Don't Forget Pairing-Friendly Curves with Odd Prime Embedding Degrees

Yu Dai[1,2], Fangguo Zhang[3,4], and Chang-an Zhao[2,4,✉]

[1] School of Mathematics, Wuhan University, Wuhan, China.
[2] School of Mathematics, Sun Yat-sen University, Guangzhou, China.
[3] School of Computer Science and Engineering, Sun Yat-sen University,Guangzhou, China.
[4] Guangdong Key Laboratory of Information Security, Guangzhou, China.
eccdaiy39@gmail.com,{isszhfg,zhaochan3}@mail.sysu.edu.cn

Abstract. Pairing-friendly curves with odd prime embedding degrees at the 128-bit security level, such as BW13-310 and BW19-286, sparked interest in the field of public-key cryptography as small sizes of the prime fields. However, compared to mainstream pairing-friendly curves at the same security level, i.e., BN446 and BLS12-446, the performance of pairing computations on BW13-310 and BW19-286 is usually considered inefficient. In this paper we investigate high performance software implementations of pairing computation on BW13-310 and corresponding building blocks used in pairing-based protocols, including hashing, group exponentiations and membership testings. Firstly, we propose efficient explicit formulas for pairing computation on this curve. Moreover, we also exploit the state-of-art techniques to implement hashing in $\mathbb{G}_1$ and $\mathbb{G}_2$, group exponentiations and membership testings. In particular, for exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$, we present new optimizations to speed up computational efficiency. Our implementation results on a 64-bit processor show that the gap in the performance of pairing computation between BW13-310 and BN446 (resp. BLS12-446) is only up to 4.9% (resp. 26%). More importantly, compared to BN446 and BLS12-446, BW13-310 is about 109.1% − 227.3%, 100% − 192.6%, 24.5% − 108.5% and 68.2% − 145.5% faster in terms of hashing to $\mathbb{G}_1$, exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, and membership testing for $\mathbb{G}_T$, respectively. These results reveal that BW13-310 would be an interesting candidate in pairing-based cryptographic protocols.

Keywords: Pairing-friendly curves · BW13-310 · high-performance software implementations

1 Introduction

A pairing on an elliptic curve E defined over a prime field $\mathbb{F}_p$ is a non-degenerate bilinear map of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three groups with the same order r. The two input groups $\mathbb{G}_1$ and $\mathbb{G}_2$ lie in $E(\mathbb{F}_{p^k})$ and the output group $\mathbb{G}_T$ is a subgroup of $\mathbb{F}_{p^k}^*$, where k is the smallest positive integer such that $r \mid p^k - 1$. Taking advantage of the powerful bilinearity property of pairings, a range of cryptographic protocols are designed, such as authenticated key agreements [12, 47], direct anonymous attestation (DAA) [10, 50] and Succinct Non-interactive ARguments of Knowledge (SNARKs) [2, 18, 19]. Very recently, pairings were also used to speed up group membership testing on several non-pairing-friendly curves [37].

The fundamental security of pairing-based protocols depends on the difficulty of solving discrete logarithm problem (DLP) on the three pairing groups. The best-known discrete logarithm algorithm on elliptic curves is the Pollard's rho algorithm [43], which requires around $\sqrt{r}$ group operations. It means that the size of r is at least 256-bit to reach the 128-bit security level. On finite fields side, the best attack algorithm is derived from the Number Field Sieve (NFS) family (eg. [44]). Before 2015, a 3072-bit full extension field $\mathbb{F}_{p^k}$ was widely believed to be 128-bit secure under the attack of NFS. With a series of new variants of NFS proposed [7,35,36], the asymptotic complexity of NFS has decreased significantly. In particular, the special extension tower number field sieve (SexTNFS) [35] is tailored to pairing-friendly fields, i.e., the characteristic p can be represented as a tiny-coefficients polynomial of moderate degree. For example, according to the estimate in [6, 32], the real security level of BN254 is around 100 $\sim$ 103-bit under the attack of SexTNFS. As a result, parameters of many pairing-friendly curves have to be re-evaluated for achieving the desired security level. In 2021, Guillevic [31] recommended a list of curves at the updated 128-bit security level. In the new estimation, the author found that BN446 and BLS12-446 are best choices for achieving the 128-bit security level in the BN and BLS families, respectively.

1.1 Pairing-friendly curves with fast exponentiation in $\mathbb{G}_1$

The development of NFS also affects the selection of pairing-friendly curves in many pairing-based cryptographic protocols. A common scenario is that a protocol is designed to minimize the workload of one party equipped with resource-constrained devices. For example, in the pairing-based DAA scheme the Trusted Platform Module (TPM) is a small discrete chip that is required to perform a few exponentiations in $\mathbb{G}_1$. One of challenges in the design of the DAA scheme is to minimize the computational cost of the TPM [50]. In this situation, pairing-friendly curves with fast exponentiation in $\mathbb{G}_1$ are very attractive. Or equivalently, curves equipped with small size of prime field are well-suited for the DAA scheme. To this aim, Clarisse *et al.* [13] recommended two 128-bit secure curves: BW13-310 with embedding degree 13 over a 310-bit field, and BW19-286 with embedding degree 19 over a 286-bit field. Besides these, there actually exist other candidates: BLS24-315 and BLS48-286. For these curves, their characteristic p can be represented by five computer words on a 64-bit processor. Among these curves, BW13-310 is competitive because of the small size of the full extension field $\mathbb{F}_{p^k}$. It is worth noting that even compared to BN446 and BLS12-446, BW13-310 also has advantage in terms of the efficiency of full extension field arithmetic. Moreover, the odd prime embedding degree k on BW13-310 also leads to a large value of $\varphi(k)$, which induces a small number of iterations for both Miller loop of the optimal pairings [49] and the group exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$ [27].

1.2 Contributions

In this paper, we give a detailed study of BW13-310. We show that this curve is a powerful candidate in pairing-based protocols at the updated 128-bit security level. Our contributions are summarized as follows:

- In Section 3, we propose a new formula for computing the optimal pairing on BW13-310. We show that the computational cost of the Miller loop comes mostly from two evaluations at the same Miller function of bit length around $\log r/(2\varphi(k))$. On this basis, we propose a shared Miller loop such that the two function evaluations can be accomplished simultaneously. In addition, we also give a slight optimization for the final exponentiation. By using these techniques, we also discuss how to compute the products of pairings on this curve in Section 4.
- In Section 5, we focus on optimizing group exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$ on BW13-310. In the case of $\mathbb{G}_2$, we show that GLV [28] and GLS [26] methods can be combined to build a $2\varphi(k)$-dimensional decomposition, which means that the number of point doublings is only around $\log r/(2\varphi(k))$ (≈ 12). In the case of $\mathbb{G}_T$, we develop an all-positive decomposition such that group inversion operation can be avoided.
- In Section 6, we provide high performance software implementations of pairing computation, hashing (to $\mathbb{G}_1$ and $\mathbb{G}_2$), group exponentiations and membership testings over BW13-310 on a 64-bit processor. By means of the RELIC cryptographic toolkit [1], detailed performance comparisons of all building blocks on BW13-310, BN446 and BLS12-446 are presented.
  - It is surprising to observe that the single pairing computation on BW13-310 is only up to 4.9% and 26% slower than that on BN446 and BLS12-446, respectively. In particular, the computation of the Miller loop on BW13-310 is even up to 48.2% faster than that on BN446. As a result, for the computation of the pairings products, BW13-310 gains an advantage over BN446, while is still slower than BLS12-446.
  - More importantly, compared to BN446 and BLS12-446, BW13-310 is about $109.1\% - 227.3\%$, $100\% - 192.6\%$, $24.5\% - 108.5\%$ and $68.2\% - 145.5\%$ faster in terms of hashing to $\mathbb{G}_1$, exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, and membership testing for $\mathbb{G}_T$, respectively.
  - On the negative side, BW13-310 also pays a penalty in terms of hashing to $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_2$.
- In Section 7, we estimate the performance of the Boneh-Lynn-Shacham (BLS) signature scheme [8] and the unbalanced Chen-Kudla (UCK) key agreement protocol [47] built on different pairing-friendly curves, including BN446, BLS12-446 and BW13-310. The results show that

- the UCK protocol built on BW13-310 is about 125.6% and 40.6% faster than that on BN446 and BLS12-446 for the resource-constrained party (Client), respectively;
- the BLS signature scheme built on BW13-310 is about both 150% faster than that on BN446 and BLS12-446 for the signer, respectively.

2 Preliminaries

Let p be a large prime, and E an ordinary elliptic curve defined by an equation of the form $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{F}_p$ are selected such that $4a^3 + 27b^2 \neq 0$. The group $E(\mathbb{F}_p)$ consists of points (x, y) satisfying the above equation with $x, y \in \mathbb{F}_p$, together with a point at infinity $\mathcal{O}$. Denote by $\#E(\mathbb{F}_p)$ the order of $E(\mathbb{F}_p)$. Then $\#E(\mathbb{F}_p)$ is precisely $p + 1 - t$, where t is the trace of the p-power Frobenius endomorphism $\pi : (x, y) \to (x^p, y^p)$. Let r be a large prime such that $r \mid \#E(\mathbb{F}_p)$. The embedding degree k with respect to r is the smallest positive integer such that $r \mid p^k - 1$. We use $\mathbb{G}_T$ to denote the subgroup of order r in $\mathbb{F}^*_{p^k}$. If $k > 1$, then the r-torsion group $E[r] = \{R \in E \mid [r]R = \mathcal{O}\}$ is contained in $E(\mathbb{F}_{p^k})$. Define $\mathbb{G}_1$ and $\mathbb{G}_2$ are eigenspaces of π acting on $E[r]$ with eigenvalues 1 and p, respectively. Or equivalently, $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$ and $\mathbb{G}_2 = E(\mathbb{F}_{p^k})[r] \cap \mathrm{Ker}(\pi\text{-}[p])$.

2.1 Optimal pairing

For any point $R \in E$ and $n \in \mathbb{Z}^+$, we denote $f_{n,R}$ as a normalized rational function with divisor

$$(f_{n,R}) = n(R) - ([n]R) - (n-1)(\mathcal{O}). \tag{1}$$

For any $i, j \in \mathbb{Z}^+$, the functions $f_{i,R}, f_{j,R}$ and $f_{i+j,R}$ satisfy the following relations:

$$f_{i+j,R} = f_{i,R} \cdot f_{j,R} \cdot \frac{\ell_{[i]R,[j]R}}{\nu_{[i+j]R}}, \tag{2}$$

where $\ell_{[i]R,[j]R}$ represents the straight line through $[i]R$ and $[j]R$, and $\nu_{[i+j]R}$ is the vertical line through $[i + j]R$. Based on Eq.(2), Miller [41] proposed a polynomial time algorithm for computing $f_{n,Q}(P)$ for any $n \in \mathbb{Z}^+$, $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, which is described in Alg. 1. Throughout this paper, we call $f_{n,R}$ as Miller function and one execution of the main loop in Alg. 1 as a basic Miller iteration.

Let m be a multiple of r with $m \nmid r^2$, and write m as $\sum_{j=0}^{\omega} c_i p^i$. An optimal pairing [49] on E is defined by

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T, (P, Q) \to \left(\prod_{\mathrm{i}=0}^{\omega} f^{p^i}_{c_i,Q}(P) \cdot \prod_{\mathrm{i}=0}^{\omega-1} \frac{\ell_{[s_{i+1}]Q,[c_i p^i]Q}(P)}{\nu_{[s_i]Q}(P)} \right)^{(p^k-1)/r}, \tag{3}$$

where $s_i = \sum_{j=i}^{\omega} c_j p^j$. Eq. (3) allows pairing evaluation to be accomplished by using around $\log r/(\varphi(k))$ basic Miller iterations and one exponentiation by $(p^k - 1)/r$.

Algorithm 1 Miller's Algorithm

```
Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $n = \sum_{i=0}^{L} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$
Output: $f_{n,Q}(P)$
T ← Q, f ← 1   ($T \leftarrow Q$, $f \leftarrow 1$)
for $i = L - 1$ down to $0$ do
    $f \leftarrow f^2 \cdot \frac{\ell_{T,T}(P)}{\nu_{[2]T}(P)}$, $T \longleftarrow 2T$
    if $n_i = 1$ then
        $f \leftarrow f \cdot \frac{\ell_{T,Q}(P)}{\nu_{T+Q}(P)}$, $T \leftarrow T + Q$
    end if
    if $n_i = -1$ then
        $f \leftarrow f \cdot \frac{\ell_{T,-Q}(P)}{\nu_{T-Q}(P)}$, $T \leftarrow T - Q$
    end if
end for
return $f$
```

For curves with even embedding degrees, pairing computation benefits from the denominator elimination optimization so that the vertical line in Alg. 1 can be ignored. Unfortunately, this technique does not apply to curves with odd prime embedding degrees. The penalty is slightly made up for by a modified Miller function $g_{m,Q}$ with divisor

$$(g_{m,Q}) = m(Q) + (-mQ) - (m+1)(\mathcal{O}). \tag{4}$$

Comparing Eqs. (1) and (4), it is easy to deduce the following relations:

$$g_{m,Q} = f_{m,Q} \cdot \nu_{[m]Q}, \tag{5}$$

$$g_{m+1,Q} = g_{m,Q} \cdot \frac{\ell_{[m]Q,Q}}{\nu_{[m]Q}}, \tag{6}$$

$$g_{2m+1,Q} = g_{m,Q}^2 \frac{\ell_{[2m]Q,Q}}{\ell_{[-m]Q,[-m]Q}}, \tag{7}$$

$$g_{4m,Q} = g_{m,Q}^4 \frac{\ell_{[2m]Q,[2m]Q}}{\ell_{[-m]Q,[-m]Q}^2}. \tag{8}$$

Exploiting Eqs. (5)-(8), Dai et al. [17] found that the optimal strategy for performing Miller loop is as follows: (1) combining two consecutive doubling steps into one quadrupling step; (2) combining one doubling and one addition steps into one doubling-addition step.

2.2 A family of curves with embedding degrees $k \equiv 1 \bmod 6$

Freeman, Scott and Teske [24, Construction 6.6] constructed a family of cyclotomic pairing-friendly curves with embedding degrees $k \equiv 1 \bmod 6$, $k \leq 1000$ and $18 \nmid k$. In particular, the characteristic p, the prime order r and the trace of Frobenius t are parameterized by

$$\begin{aligned} r(z) &= \Phi_{6k}(z), \\ p(z) &= \tfrac{1}{3}(z+1)^2(z^{2k} - z^k + 1) - z^{2k+1}, \\ t(z) &= -z^{k+1} + z + 1, \end{aligned}$$

where $\Phi_l(\cdot)$ represents l-th cyclotomic polynomial. All members in this family have j-invariant 0 and are defined by an equation of the form $y^2 = x^3 + b$ for some $b \in \mathbb{F}_p^*$. Following [13], this family is named as the BW family. By the form of $r(z)$, we have $r(z) \mid (z^{2k} - z^k + 1)$, which implies that

$$\begin{aligned} z^2 - z \cdot p(z) + p^2(z) &\equiv z^2 + z \cdot z^{2k+1} + z^{4k+2} \\ &\equiv z^2 \cdot (1 + z^{2k} + z^{4k}) \\ &\equiv 0 \bmod r(z). \end{aligned}$$

Thus, one of short vectors $(c_0, c_1, \cdots, c_\omega)$ for the optimal pairing in this family is given by $(z^2, -z, 1, 0, \cdots, 0)$. Plugging this vector into Eq. (3), the corresponding formula of the optimal pairing is

$$e(P,Q) = (f_{z^2,Q}(P) \cdot f_{-z,Q}^{p}(P) \cdot \ell_{\pi^2(Q),\pi([-z]Q)}(P))^{(p^k-1)/r}. \tag{9}$$

It is known from [21, Lemma 3.5] that

$$f_{z^2,Q} = f_{-z,Q}^{-z} \cdot f_{-z,[-z]Q}.$$

Therefore, Eq. (9) can be rewritten as

$$e(P,Q) = \left(f_{-z,Q}^{-z+p}(P) \cdot f_{-z,[-z]Q}(P) \cdot \ell_{\pi^2(Q),\pi([-z]Q)}(P)\right)^{(p^k-1)/r}. \tag{10}$$

BW13-310 and BW19-286: In the BW family, BW13-310 and BW19-286 are the two curves defined by setting $k = 13$ and 19, $z = -2224$ and -145, $b = -17$ and 31, respectively. In particular, the selected prime p on BW13-310 satisfies that $p \equiv 1 \bmod 13$, so that the full extension field $\mathbb{F}_{p^{13}}$ can be represented as $\mathbb{F}_p[v]/(v^{13} - \alpha)$ for some $\alpha \in \mathbb{F}_p^*$. Using Magma [9], it is easy to check that the polynomial $v^{13} - 2$ is irreducible over $\mathbb{F}_p$, which means that we can select the value of α as 2. According to the estimation in [13, 31], both curves are 128-bit secure even under the attack of the SexTNFS.

3 Single Pairing Computation on BW13-310

Notations. We denote by $\mathbf{a}$, $\mathbf{m}$, $\mathbf{m}_u$, $\mathbf{s}$, $\mathbf{s}_u$, $\mathbf{i}$ and $\mathbf{r}$ the computational costs of addition, multiplication, multiplication without reduction, squaring, squaring without reduction, inversion and modular reduction in $\mathbb{F}_p$, respectively. It is obvious that $\mathbf{m} = \mathbf{m}_u + \mathbf{r}$ and $\mathbf{s} = \mathbf{s}_u + \mathbf{r}$. Likewise, we use $\tilde{\mathbf{a}}$, $\tilde{\mathbf{m}}$, $\tilde{\mathbf{m}}_u$ $\tilde{\mathbf{s}}$, $\tilde{\mathbf{s}}_u$, $\tilde{\mathbf{i}}$, $\tilde{\mathbf{f}}$, $\tilde{\mathbf{e}}$ and $\tilde{\mathbf{r}}$ to represent the computational costs of addition, multiplication, multiplication without reduction, squaring, squaring without reduction, inversion, Frobenius, exponentiation by $|z|$ and modular reduction in $\mathbb{F}_{p^{13}}$, respectively. The cyclotomic group $\mathbb{G}_{\Phi_{13}(p)}$ is defined by $\mathbb{G}_{\Phi_{13}(p)} = \{\beta \in \mathbb{F}_{p^{13}} \mid \beta^{\Phi_{13}(p)} = 1\}$. We denote by $\tilde{\mathbf{i}}_c$ the cost of the inversion in $\mathbb{G}_{\Phi_{13}(p)}$. The notation $\times$ is used to denote field multiplication without reduction. For any point Q, we use (x_Q, y_Q) to represent the point in affine coordinates, and (X_Q, Y_Q, Z_Q) in jacobian coordinates, which means that $x_Q = X_Q/Z_Q^2$ and $y_Q = Y_Q/Z_Q^3$. Given a vector $\mathbf{n} = (n_0, n_1, \cdots, n_m)$, the notation $\|\mathbf{n}\|_\infty$ represents $\max\{|n_0|, |n_1|, \cdots, |n_m|\}$.

In this section, we propose a new formula for pairing computation in the BW family, and discuss how to apply it to BW13-310 in detail. Since $p \equiv 1 \bmod 3$, there exists an endomorphism $\phi : (x, y) \to (\omega \cdot x, y)$ such that $\phi(Q) = [\lambda]Q$ for any $Q \in \mathbb{G}_2$, where ω is a cube primitive root of unity in $\mathbb{F}_p^*$ and λ is a root of the quadratic congruence equation $x^2 + x + 1 \equiv 0 \bmod r$. Recall that

$$z^{2k} - z^k + 1 \equiv 0 \bmod r,$$

which implies that λ is $-z^k$ or $z^k - 1$. Fix the parameter ω such that $\lambda = -z^k$ and define $\psi = \pi \circ \phi$. Then, we have

$$\psi(Q) = [\lambda]\pi(Q) = [\lambda \cdot (t-1) \bmod r]Q = [-z]Q. \tag{11}$$

Based on this observation, we prove the following theorem.

Theorem 1. *Let notation as above. Then the formula of the optimal pairing in the* BW *family can be expressed as*

$$e(Q, P) = \left(f_{-z,Q}^{-z+p}(P) \cdot f_{-z,Q}^{p}(\hat{\phi}(P)) \cdot \ell_{\pi^2(Q), \pi^2 \circ \phi(Q)}(P)\right)^{(p^k-1)/r}, \tag{12}$$

where $\hat{\phi} = \phi^2$.

Proof. By [33, Lemma 3] and Eq. (11), it is easy to see that

$$f_{-z,[-z]Q} = f_{-z,\pi\circ\phi(Q)} = f_{-z,\phi(Q)}^{p}. \tag{13}$$

By Eq. (1), we obtain that

$$(f_{-z,\phi(Q)}) = -z(\phi(Q)) - (\phi([-z]Q)) - (-z-1)(\mathcal{O}).$$

Since ϕ is an automorphism on E, we have

$$\phi^*(f_{-z,\phi(Q)}) = -z(Q) - ([-z]Q) - (-z-1)(\mathcal{O}) = (f_{-z,Q}), \tag{14}$$

where ϕ^* is the pullback of ϕ [25, Definition 8.3.1]. Furthermore, since

$$\phi^*(f_{-z,\phi(Q)}) = (f_{-z,\phi(Q)} \circ \phi),$$

Eq. (14) implies that

$$f_{-z,\phi(Q)} = f_{-z,\phi(Q)} \circ \phi \circ \hat{\phi} = f_{-z,Q} \circ \hat{\phi}. \tag{15}$$

By the fact that $p \equiv 1 \bmod 3$ and $\hat{\phi}^3 = 1$ we get

$$\hat{\phi}^p = \hat{\phi}. \tag{16}$$

Combining Eqs.(13), (15) and (16), it yields

$$f_{-z,[-z]Q} = f^p_{-z,\phi(Q)} = f^p_{-z,Q} \circ \hat{\phi}^p = f^p_{-z,Q} \circ \hat{\phi}. \tag{17}$$

Inserting Eq. (17) into Eq. (10) and replacing $[-z]Q$ by $\psi(Q)$, we complete the proof of this theorem.

In Theorem 1, we propose a new formula for computing the optimal pairing in the BW family, which is suitable for BW13-310 and BW19-286. Using the new formula, the number of basic Miller iterations is reduced to $\lfloor \log|z| \rfloor \approx \log r/(2\varphi(k))$. In detail, when performing Miller's algorithm using Eq. (12), the computational cost largely comes from two evaluations at the same Miller function of length $\log|z|$, i.e., $f_{-z,Q}(P)$ and $f_{-z,Q}(\hat{\phi}(P))$. Recently, Fouotsa et al. [23] proposed the x-superoptimal pairing on BW13-310 and BW19-286, which is expressed as

$$a^x_{sup}(Q,P) = \Big((f_{-z,Q}(P) \cdot f^{-1}_{-z,Q}(\hat{\phi}(P)))^{-z} \cdot (f^{-1}_{-z,Q}(\hat{\phi}(P)) \cdot f_{-z,Q}(\phi(P)))^p \Big)^{(p^{13}-1)/r}. \tag{18}$$

Clearly, Eqs. (12) and (18) require the same number of iterations when performing Miller's algorithm. But the latter requires three evaluations at the same Miller function of length $\log|z|$, i.e., $f_{-z,Q}(P)$, $f_{-z,Q}(\phi(P))$ and $f_{-z,Q}(\hat{\phi}(P))$. Therefore, compared to the x-superoptimal pairing, our proposed formula would be more efficient. In the following, we investigate how to perform the pairing computation on BW13-310 in detail.

3.1 Shared Miller loop

Computing a single pairing by multiple Miller function evaluations were studied in [3,45,52]. This inspires us to consider how to speed up pairing computation on BW13-310 by using Eq. (12), i.e., computing $f_{-z,Q}(P)$ and $f_{-z,Q}(\hat{\phi}(P))$. Since the points P and $\hat{\phi}(P)$ have the same y-coordinates, the two Miller function evaluations share a large amount of intermediate values during Miller iteration. Hence, it would be efficient to calculate $f_{-z,Q}(P)$ and $f_{-z,Q}(\hat{\phi}(P))$ simultaneously. In other words, we can accomplish two Miller function evaluations at a shared Miller loop. Recall from Section 2 that the seed $z = -2224$ on BW13-310, and the Miller function $f_{2224,Q}$ can be obtained from $f_{1,Q}$ via the following sequence:

$$f_{1,Q} \to g_{1,Q} \to g_{4,Q} \to g_{16,Q} \to g_{17,Q} \to g_{68,Q} \to g_{69,Q} \to g_{139,Q} \to g_{556,Q} \to g_{2224,Q} \to f_{2224,Q}. \tag{19}$$

For any $i \in \mathbb{Z}$, we denote by $N_{i,Q}$ and $D_{i,Q}$ the numerator and denominator of $g_{i,Q}$, respectively. According to Eq. (5), we have

$$g_{1,Q}(x,y) = f_{1,Q}(x,y) \cdot \nu_{1,Q}(x,y) = x - x_Q$$

for any point $(x,y) \in E$. Therefore, it is natural to set

$$N_{1,Q}(P) = x_P - x_Q, D_{1,Q}(P) = 1, N_{1,Q}(\hat{\phi}(P)) = \tilde{x}_P - x_Q, D_{1,Q}(\hat{\phi}(P)) = 1, \tag{20}$$

where $\tilde{x}_P$ represents the x-coordinate of $\hat{\phi}(P)$. In the following we will discuss how to update the terms $N_{m,Q}(P)$, $D_{m,Q}(P)$, $N_{m,Q}(\hat{\phi}(P))$ and $D_{m,Q}(\hat{\phi}(P))$ in a shared Miller loop for any $m \in \mathbb{Z}$. Before that, we use $T = (X_T, Y_T, Z_T)$ to denote $[m]Q$ in Jacobian coordinates.

Shared addition step(SADD) In this subsection we show how to obtain $N_{m+1,Q}(P)$, $D_{m+1,Q}(P)$, $N_{m+1,Q}(\hat{\phi}(P))$ and $D_{m+1,Q}(\hat{\phi}(P))$ from $N_{m,Q}(P)$, $D_{m,Q}(P)$, $N_{m,Q}(\hat{\phi}(P))$ and $D_{m,Q}(\hat{\phi}(P))$, respectively. To this end, the point $T+Q$ is first calculated. Since T and Q are represented in Jacobian and affine coordinates respectively, we adopt the mixed addition formula presented in [5, Section 4.3.2] to compute $T+Q$, which is given by

$$\alpha_{T+Q} = y_Q \cdot Z_T^3 - Y_T, \beta_{T+Q} = x_Q \cdot Z_T^2 - X_T, X_{T+Q} = \alpha_{T+Q}^2 - 2X_T \cdot \beta_{T+Q}^2 - \beta_{T+Q}^3,$$
$$Y_{T+Q} = \alpha_{T+Q} \cdot (X_T \cdot \beta_{T+Q}^2 - X_{T+Q}) - Y_T \cdot \beta_{T+Q}^3, Z_{T+Q} = Z_T \cdot \beta_{T+Q}.$$

It can be done by using the following sequence of operations:

$$A = Z_T^2, B = A \cdot Z_T, C = B \cdot y_Q - Y_T, D = A \cdot x_Q - X_T, E = D^2, F = D \cdot E, G = X_T \cdot E,$$
$$X_{T+Q} = C^2 - 2G - F, U_0 = C \times (G - X_{T+Q}), U_1 = Y_T \times F, Y_{T+Q} = (U_0 - U_1) \bmod p,$$
$$Z_{T+Q} = Z_T \cdot D.$$

The above calculation comes at a cost of $6\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{r}} + 8\tilde{\mathbf{a}}$, assuming that computing $U_0 - U_1$ requires $2\tilde{\mathbf{a}}$. For any point (x,y), it can be deduced from Eq. (6) that

$$\begin{aligned} Nm+1, Q(x,y) &= N_{m,Q}(x,y) \cdot L_{T,a,1}(x,y), \\ D_{m+1,Q}(x,y) &= D_{m,Q}(x,y) \cdot L_{T,a,2}(x,y), \end{aligned} \tag{21}$$

where $L_{T,a,1}(x,y)$ and $L_{T,a,2}(x,y)$ are given by

$$\begin{aligned} L_{T,a,1}(x,y) &= \beta_{T+Q} \cdot (y \cdot Z_T^3 - Y_T) - \alpha_{T+Q} \cdot (x \cdot Z_T^2 - X_T), \\ L_{T,a,2}(x,y) &= Z_{T+Q} \cdot (x \cdot Z_T^2 - X_T). \end{aligned}$$

Since α_{T+Q}, β_{T+Q}, Z_{T+Q}, Z_T^2 and Z_T^3 have been obtained at the point addition step, we perform the following sequence of operations to compute $L_{T,a,1}(P)$,

$L_{T,a,2}(P)$, $L_{T,a,1}(\hat{\phi}(P))$ and $L_{T,a,2}(\hat{\phi}(P))$ which requires $2\tilde{\mathbf{m}} + 3\tilde{\mathbf{m}}_u + 39\mathbf{m} + 2\tilde{\mathbf{r}} + 7\tilde{\mathbf{a}}$ as

$$A = y_P \cdot Z_T^3 - Y_T, B = x_P \cdot Z_T^2 - X_T, C = \tilde{x}_P \cdot Z_T^2 - X_T, U_0 = \beta_{T+Q} \times A, U_1 = \alpha_{T+Q} \times B,$$
$$U_2 = \alpha_{T+Q} \times C, L_{a,1}(P) = (U_0 - U_1) \bmod p, L_{T,a,1}(\hat{\phi}(P)) = (U_0 - U_2) \bmod p,$$
$$L_{T,a,2}(P) = Z_{T+Q} \cdot B, L_{T,a,2}(\hat{\phi}(P)) = Z_{T+Q} \cdot C.$$

On this basis, we can obtain $N_{m+1,Q}(P)$, $D_{m+1,Q}(P)$, $N_{m+1,Q}(\hat{\phi}(P))$ and $D_{m+1,Q}(\hat{\phi}(P))$ from Eq. (21) at a cost of $4\tilde{\mathbf{m}}$. In summary, the computational cost at the SADD step is

$$\underbrace{6\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{r}} + 8\tilde{\mathbf{a}}}_{\text{point addition}} + \underbrace{2\tilde{\mathbf{m}} + 3\tilde{\mathbf{m}}_u + 39\mathbf{m} + 2\tilde{\mathbf{r}} + 7\tilde{\mathbf{a}}}_{L_{T,a,1} \text{ and } L_{T,a,2}} + \underbrace{4\tilde{\mathbf{m}}}_{\text{the final step}}$$
$$= 12\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + 39\mathbf{m} + 3\tilde{\mathbf{r}} + 15\tilde{\mathbf{a}}.$$

Shared doubling-addition step(SDBLADD) By combining one doubling and one addition steps in the shared Miller loop, we can efficiently obtain $N_{2m+1,Q}(P)$, $D_{2m+1,Q}(P)$, $N_{2m+1,Q}(\hat{\phi}(P))$ and $D_{2m+1,Q}(\hat{\phi}(P))$ from $N_{m,Q}(P)$, $D_{m,Q}(P)$, $N_{m,Q}(\hat{\phi}(P))$ and $D_{m,Q}(\hat{\phi}(P))$, respectively. Firstly, using the formula presented in [5, Section 4.3.1], the point $2T = (X_{2T}, Y_{2T}, Z_{2T})$ is given by

$$X_{2T} = \frac{9}{4}X_T^4 - 2X_T \cdot Y_T^2, Y_{2T} = \frac{3}{2}X_T^2 \cdot (X_T \cdot Y_T^2 - X_{2T}) - Y_T^4, Z_{2T} = Y_T \cdot Z_T.$$

Thus the computation of point doubling requires $2\tilde{\mathbf{m}} + \tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + \tilde{\mathbf{r}} + 7\tilde{\mathbf{a}}$ using the following sequence of operations:

$$A = X_T^2, B = A/2, C = A + B, D = C^2, E = Y_T^2, F = X_T \cdot E, X_{2T} = D - 2F, G = F - X_{2T},$$
$$U_0 = C \times G, U_1 = E \times E, Y_{2T} = (U_0 - U_1) \bmod p, Z_{2T} = Y_T \cdot Z_T.$$

On this basis, one can obtain the point $2T + Q$ via one mixed point addition:

$$\alpha_{2T+Q} = y_Q \cdot Z_{2T}^3 - Y_{2T}, \beta_{2T+Q} = x_Q \cdot Z_{2T}^2 - X_{2T}, X_{2T+Q} = \alpha_{2T+Q}^2 - 2X_{2T} \cdot \beta_{2T+Q}^2 - \beta_{2T+Q}^3,$$
$$Y_{2T+Q} = \alpha_{2T+Q} \cdot (X_{2T} \cdot \beta_{2T+Q}^2 - X_{2T+Q}) - Y_{2T} \cdot \beta_{2T+Q}^3, Z_{2T+Q} = Z_{2T} \cdot \beta_{2T+Q}.$$

From Eq. (7), we deduce that

$$\begin{aligned} N_{2m+1,Q}(x,y) &= N_{m,Q}^2(x,y) \cdot L_{T,d,1}(x,y), \\ D_{2m+1,Q}(x,y) &= D_{m,Q}^2(x,y) \cdot L_{T,d,2}(x,y). \end{aligned} \tag{22}$$

where $L_{T,d,1}(x,y)$ and $L_{T,d,2}(x,y)$ are given by

$$L_{T,d,1}(x,y) = \beta_{2T+Q} \cdot (y \cdot Z_{2T}^3 - Y_{2T}) - \alpha_{2T+Q} \cdot (x \cdot Z_{2T}^2 - X_{2T}),$$
$$L_{T,d,2}(x,y) = \beta_{2T+Q} \cdot (y \cdot Z_{2T}^3 - Y_{2T}) + \frac{3}{2}X_T^2 \cdot \beta_{2T+Q} \cdot (x \cdot Z_{2T}^2 - X_{2T}).$$

Since the values of $\frac{3}{2}X_T^2$, Z_{2T}^2, Z_{2T}^3, α_{2T+Q} and β_{2T+Q} have been obtained, the computation of $L_{T,d,1}(P)$, $L_{T,d,2}(P)$, $L_{T,d,1}(\hat{\phi}(P))$ and $L_{T,d,2}(\hat{\phi}(P))$ can be done in $\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 4\tilde{\mathbf{r}} + 39\mathbf{m} + 11\tilde{\mathbf{a}}$ as follows:

$$\begin{aligned}
&A = y_P \cdot Z_{2T}^3 - Y_{2T}, B = x_P \cdot Z_{2T}^2 - X_{2T}, C = \tilde{x}_P \cdot Z_{2T}^2 - X_{2T}, D = \frac{3}{2}X_T^2 \cdot \beta_{2T+Q},\\
&U_0 = \beta_{2T+Q} \times A, U_1 = \alpha_{2T+Q} \times B, U_2 = \alpha_{2T+Q} \times C, U_3 = D \times B, U_4 = D \times C,\\
&L_{T,d,1}(P) = (U_0 - U_1) \bmod p, L_{T,d,2}(P) = (U_0 + U_3) \bmod p,\\
&L_{T,d,1}(\hat{\phi}(P)) = (U_0 - U_2) \bmod p, L_{T,d,2}(\hat{\phi}(P)) = (U_0 + U_4) \bmod p.
\end{aligned}$$

Finally, it can be seen from Eq. (22) that the computation of $N_{2m+1,Q}(P)$, $D_{2m+1,Q}(P)$, $N_{2m+1,Q}(\hat{\phi}(P))$ and $D_{2m+1,Q}(\hat{\phi}(P))$ requires $4\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}}$. In total, the computational cost at the SDBLADD step is

$$\begin{aligned}
&\underbrace{2\tilde{\mathbf{m}} + \tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + \tilde{\mathbf{r}} + 7\tilde{\mathbf{a}}}_{\text{point doubling}} + \underbrace{6\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{r}} + 8\tilde{\mathbf{a}}}_{\text{point addition}}\\
&+ \underbrace{\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 4\tilde{\mathbf{r}} + 39\mathbf{m} + 11\tilde{\mathbf{a}}}_{L_{T,d,1} \text{ and } L_{T,d,2}} + \underbrace{4\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}}}_{\text{the final step}}\\
&= 13\tilde{\mathbf{m}} + 8\tilde{\mathbf{m}}_u + 10\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 39\mathbf{m} + 26\tilde{\mathbf{a}}.
\end{aligned}$$

Shared quadrupling step(SQPL) By combining two doubling steps into one quadrupling step in the shared Miller loop, we can efficiently perform the following four function updates:

$$\begin{aligned}
&N_{4m,Q}(P) \leftarrow N_{m,Q}(P), D_{4m,Q}(P) \leftarrow D_{m,Q}(P),\\
&N_{4m,Q}(\hat{\phi}(P)) \leftarrow N_{m,Q}(\hat{\phi}(P)), D_{4m,Q}(\hat{\phi}(P)) \leftarrow D_{m,Q}(\hat{\phi}(P)).
\end{aligned}$$

We first perform two successive point doublings to calculate $4T = (X_{4T}, Y_{4T}, Z_{4T})$. It can be seen from Section 3.1.2 that it costs $4\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 6\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 2\tilde{\mathbf{r}} + 14\tilde{\mathbf{a}}$. Then, straightforward computation using (8) reveals that

$$N_{4m,Q}(x,y) = N_{m,Q}^4 \cdot L_{T,q,1}(x,y), D_{4m,Q}(x,y) = D_{m,Q}^4 \cdot \left(L_{T,q,2}(x,y)\right)^2, \quad (23)$$

where $L_{T,q,1}$ and $L_{T,q,2}$ are given by

$$\begin{aligned}
&L_{T,q,1}(x,y) = Z_{4T} \cdot Z_{2T}^2 \cdot \left(y \cdot Z_{4T} \cdot Z_{2T}^2 - \frac{3}{2}X_{2T}^2 \cdot (x \cdot Z_{2T}^2 - X_{2T}) - Y_{2T}^2\right),\\
&L_{T,q,2}(x,y) = y \cdot Z_{4T} \cdot Z_{2T}^2 + \frac{3}{2}X_T^2 \cdot Y_{2T} \cdot (x \cdot Z_{2T}^2 - X_{2T}) - Y_{2T}^2.
\end{aligned}$$

During the procedure of point quadrupling, the values of $\frac{3}{2}X_T^2$, $\frac{3}{2}X_{2T}^2$ and Y_{2T}^2 can be obtained. Then, we compute $L_{T,q,1}(P)$, $L_{T,q,2}(P)$, $L_{T,q,1}(\hat{\phi}(P))$ and

$L_{T,q,2}(\hat{\phi}(P))$ using the following sequence of operations:

$$A=Z_{2T}^2, B=Z_{4T}\cdot A, C=x_P\cdot A-X_{2T}, D=\tilde{x}_P\cdot A-X_{2T}, E=\frac{3}{2}X_T^2\cdot Y_{2T}, U_0=y_P\times B,$$
$$U_1=\frac{3}{2}X_T^2\times C, U_2=\frac{3}{2}X_T^2\times D, U_3=C\times E, U_4=D\times E, F=(U_0-U_1)\bmod p,$$
$$G=(U_0-U_2)\bmod p, H=(U_0+U_3)\bmod p, I=(U_0+U_4)\bmod p, L_{T,q,1}(P)=B\cdot(F-Y_{2T}^2),$$
$$L_{T,q,2}(P)=H-Y_{2T}^2, L_{T,q,1}(\hat{\phi}(P))=B\cdot(G-Y_{2T}^2), L_{T,q,2}(\hat{\phi}(P))=I-Y_{2T}^2.$$

The above computation costs $4\tilde{\mathbf{m}}+4\tilde{\mathbf{m}}_u+\tilde{\mathbf{s}}+4\tilde{\mathbf{r}}+26\mathbf{m}+13\mathbf{m}_u+14\tilde{\mathbf{a}}$. At last, we can obtain $N_{4m,Q}(P)$, $D_{4m,Q}(P)$, $N_{4m,Q}(\hat{\phi}(P))$ and $D_{4m,Q}(\hat{\phi}(P))$ from Eq. (23) at a cost of $4\tilde{\mathbf{m}}+8\tilde{\mathbf{s}}$. In total, the computational cost at the SQPL step is

$$\underbrace{4\tilde{\mathbf{m}}+2\tilde{\mathbf{m}}_u+6\tilde{\mathbf{s}}+2\tilde{\mathbf{s}}_u+2\tilde{\mathbf{r}}+14\tilde{\mathbf{a}}}_{\text{point quadrupling}}+\underbrace{4\tilde{\mathbf{m}}+4\tilde{\mathbf{m}}_u+\tilde{\mathbf{s}}+4\tilde{\mathbf{r}}+26\mathbf{m}+13\mathbf{m}_u+14\tilde{\mathbf{a}}}_{L_{T,q,1}\text{ and }L_{T,q,2}}+\underbrace{4\tilde{\mathbf{m}}+8\tilde{\mathbf{s}}}_{\text{the final step}}$$
$$=12\tilde{\mathbf{m}}+6\tilde{\mathbf{m}}_u+15\tilde{\mathbf{s}}+2\tilde{\mathbf{s}}_u+6\tilde{\mathbf{r}}+26\mathbf{m}+13\mathbf{m}_u+28\tilde{\mathbf{a}}.$$

In addition, it can be seen from Eq. (20) that when $m=1$, both $D_{m,Q}(P)$ and $D_{m,Q}(\hat{\phi}(P))$ are equal to 1, which indicates that two full extension field squarings can be saved.

Function transformation According to the relation between $f_{-z,Q}$ and $g_{-z,Q}$, we immediately have

$$\begin{aligned} f_{-z,Q}(P)&=\frac{g_{-z,Q}(P)}{\nu_{\psi(Q)}(P)}=\frac{N_{-z,Q}(P)}{D_{-z,Q}(P)\cdot(x_P-\omega\cdot x_Q^p)},\\ f_{-z,Q}(\hat{\phi}(P))&=\frac{g_{-z,Q}(\hat{\phi}(P))}{\nu_{\psi(Q)}(\hat{\phi}(P))}=\frac{N_{-z,Q}(\hat{\phi}(P))}{D_{-z,Q}(\hat{\phi}(P))\cdot(\tilde{x}_P-\omega\cdot x_Q^p)}. \end{aligned}\tag{24}$$

Moreover, the points $\pi^2(Q)$ and $\pi^2(\phi(Q))$ have the same x-coordinates, that is,

$$\ell_{\pi^2(Q),\pi^2(\phi(Q))}(P)=y_p-y_Q^{p^2}.\tag{25}$$

Putting Eqs.(24) and (25) together, Eq. (12) can be rewritten as $e(P,Q)=(\frac{L_1}{L_2})^{(p^{13}-1)/r}$, where

$$\begin{aligned} L_1&=\Big(\frac{N_{-z,Q}(P)}{D_{-z,Q}(P)\cdot(x_P-\omega\cdot x_Q^p)}\Big)^{-z+p}\cdot\big(N_{-z,Q}(\hat{\phi}(P))\big)^p\cdot(y_p-y_Q^{p^2}),\\ L_2&=\big(D_{-z,Q}(\hat{\phi}(P))\cdot(\tilde{x}_P-\omega\cdot x_Q^p)\big)^p. \end{aligned}$$

Once the values of $N_{-z,Q}(P)$, $D_{-z,Q}(P)$, $N_{-z,Q}(\hat{\phi}(P))$ and $D_{-z,Q}(\hat{\phi}(P))$ are given, the computation of L_1 and L_2 can be done at a cost of $\tilde{\mathbf{e}}+\tilde{\mathbf{i}}+6\tilde{\mathbf{m}}+13\mathbf{m}+5\tilde{\mathbf{f}}+3\tilde{\mathbf{a}}$. We will delay the inversion of L_2 into the easy part of the final exponentiation such that one inversion can be saved.

3.2 The final exponentiation

An optimized final exponentiation routine is critical for fast pairing computation on BW13-310. The exponent $(p^{13-1})/r$ can be split as

$$(p^{13-1})/r = \underbrace{(p-1)}_{\text{easy part}} \cdot \underbrace{(1+p+p^2+\cdots+p^{12})/r}_{\text{hard part}} \cdot$$

Raising L_1/L_2 to the power of $p-1$ is easy, which can be done at a cost of $\tilde{\mathbf{i}} + 3\tilde{\mathbf{m}} + 2\tilde{\mathbf{f}}$ as follows:

$$f = (L_1/L_2)^{p-1} = \frac{L_1^p \cdot L_2}{L_2^p \cdot L_1}.$$

The bottleneck of the final exponentiation is to raise f to the power of the hard part. In [17], the exponent of the hard part can be replaced by

$$h = \lambda_0 + 3 \cdot p + \big(\sum_{i=1}^{3} \lambda_i \cdot p^{i-1}\big) \cdot \big(\sum_{i=0}^{3} x^{3i} \cdot p^{10-3i}\big),$$

where $x = -z$, and λ_0, λ_1, λ_2 and λ_3 are given by

$$\begin{aligned}
\lambda_0 &= -x^{15} - 2x^{14} - 2x^{13} - x^{12} - x^2 + 2x + 2,\\
\lambda_1 &= -x^{18} - 2x^{17} - 2x^{16} - x^{15} - x^5 + 2x^4 + 2x^3,\\
\lambda_2 &= x^{16} + x^{15} + x^{14} + x^4 + 2x^3 - x^2 + x,\\
\lambda_3 &= x^{16} + x^{15} + x^{14} - 4x^2 - x - 1.
\end{aligned}$$

However, unlike the case of even embedding degrees, the cost of the inversion of f is still expensive. In order to avoid this operation as much as possible, we can break λ_i as $\lambda_{i,0} - \lambda_{i,1}$ for $i = 0, 1, 2, 3$, where

$$\begin{aligned}
\lambda_{0,0} &= 2x + 2, & \lambda_{0,1} &= x^{15} + 2x^{14} + 2x^{13} + x^{12} + x^2,\\
\lambda_{1,0} &= 2x^4 + 2x^3, & \lambda_{1,1} &= x^{18} + 2x^{17} + 2x^{16} + x^{15} + x^5,\\
\lambda_{2,0} &= x^{16} + x^{15} + x^{14} + x^4 + 2x^3 + x, & \lambda_{2,1} &= x^2,\\
\lambda_{3,0} &= x^{16} + x^{15} + x^{14}, & \lambda_{3,1} &= 4x^2 + x + 1.
\end{aligned}$$

The above exponents can be classified by degree into the following two categories:

$$\text{low degrees} : \lambda_{0,0}, \lambda_{1,0}, \lambda_{2,1} \text{ and } \lambda_{3,1}; \text{ high degrees} : \lambda_{0,1}, \lambda_{1,1}, \lambda_{2,0} \text{ and } \lambda_{3,0}.$$

Firstly, the values of $f^{\lambda_{0,0}}$, $f^{\lambda_{1,0}}$, $f^{\lambda_{2,1}}$ and $f^{\lambda_{3,1}}$ can be computed using the following operations:

$$f \to f^3 \to f^x \to f^x \cdot f \to f^{2(x+1)} \to f^{x^2} \to f^{4x^2} \cdot f^{x+1} \to f^{x^3} \to f^{x^4} \to f^{x^3} \cdot f^{x^4} \to f^{2(x^3+x^4)}. \tag{26}$$

Since only one additional field multiplication is required for obtaining f^3 at the process of computing f^x, the cost of computing (26) is $4\tilde{\mathbf{e}} + 4\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}}$. Denote g by $f^{(x^{12}+x^{13}+x^{14})}$. On the basis of (26), we then compute g with $10\tilde{\mathbf{e}} + \tilde{\mathbf{m}}$:

$$f^{x^5} \to f^{x^3+x^4} \cdot f^{x^5} \to f^{x^9(x^3+x^4+x^5)}.$$

To compute $f^{\lambda_{0,1}}$, $f^{\lambda_{1,1}}$, $f^{\lambda_{2,0}}$ and $f^{\lambda_{3,0}}$, the following operations are performed:

$$g \to g^x \to g \cdot g^x \cdot f^{x^2} \to g^{x^2} \to g^{x^2} \cdot f^{x^3+x^4} \cdot f^{x^3} \cdot f^x \to g^{x^3} \to g^{x^4} \to g^{x^3} \cdot g^{x^4} \cdot f^{x^5}. \tag{27}$$

The cost of computing (27) is $4\tilde{\mathbf{e}} + 7\tilde{\mathbf{m}}$. Using the trick of Montgomery's simultaneous inversion [42], we then can compute the terms $f_0 = f^{\lambda_0}$ and $f_1 = f^{\lambda_1+\lambda_2\cdot p+\lambda_3\cdot p^2}$ as follows:

$$\begin{aligned} f_0 &= \frac{f^{\lambda_{0,0}}}{f^{\lambda_{0,1}}} = \frac{f^{\lambda_{0,0}} \cdot (f^{\lambda_{1,1}} \cdot f^{\lambda_{2,1}\cdot p} \cdot f^{\lambda_{3,1}\cdot p^2})}{f^{\lambda_{0,1}} \cdot (f^{\lambda_{1,1}} \cdot f^{\lambda_{2,1}\cdot p} \cdot f^{\lambda_{3,1}\cdot p^2})}, \\ f_1 &= \frac{f^{\lambda_{1,0}} \cdot f^{\lambda_{2,0}\cdot p} \cdot f^{\lambda_{3,0}\cdot p^2}}{f^{\lambda_{1,1}} \cdot f^{\lambda_{2,1}\cdot p} \cdot f^{\lambda_{3,1}\cdot p^2}} = \frac{f^{\lambda_{0,1}} \cdot (f^{\lambda_{1,0}} \cdot f^{\lambda_{2,0}\cdot p} \cdot f^{\lambda_{3,0}\cdot p^2})}{f^{\lambda_{0,1}} \cdot (f^{\lambda_{1,1}} \cdot f^{\lambda_{2,1}\cdot p} \cdot f^{\lambda_{3,1}\cdot p^2})}, \end{aligned}$$

which requires $\tilde{\mathbf{i}}_c + 9\tilde{\mathbf{m}} + 4\tilde{\mathbf{f}}$. Finally, raising f to the power of h can be expressed as

$$f_0 \cdot f^{3\cdot p} \cdot f_1^{p^{10}+x^3\cdot p^7+x^6\cdot p^4+x^9\cdot p} = f_0 \cdot (f^3 \cdot f_1^{x^9})^p \cdot f_1^{p^{10}+x^3\cdot p^7+x^6\cdot p^4}. \tag{28}$$

Since the value of f^3 can be obtained from (26), the computation of (28) requires $9\tilde{\mathbf{e}} + 5\tilde{\mathbf{m}} + 4\tilde{\mathbf{f}}$.

3.3 Operation counts

$\mathbb{F}_{p^{13}}$ Arithmetic	Operation Counts in $\mathbb{F}_p$
$\tilde{\mathbf{a}}$	$13\mathbf{a}$
$\tilde{\mathbf{m}}$	$66\mathbf{m}_u + 502\mathbf{a} + 13\mathbf{r}$
$\tilde{\mathbf{m}}_u$	$66\mathbf{m}_u + 502\mathbf{a}$
$\tilde{\mathbf{s}}$	$66\mathbf{s}_u + 443\mathbf{a} + 13\mathbf{r}$
$\tilde{\mathbf{s}}_u$	$66\mathbf{s}_u + 443\mathbf{a}$
$\tilde{\mathbf{r}}$	$13\mathbf{r}$
$\tilde{\mathbf{i}}$	$277\mathbf{m}_u+73\mathbf{m}+\mathbf{i}+2034\mathbf{a}+53\mathbf{r}$
$\tilde{\mathbf{i}}_c$	$264\mathbf{m}_u + 60\mathbf{m} + 2008\mathbf{a} + 52\mathbf{r}$
$\tilde{\mathbf{e}}$	$198\mathbf{m}_u + 726\mathbf{s}_u + 6379\mathbf{a} + 182\mathbf{r}$
$\tilde{\mathbf{f}}$	$12\mathbf{m}$

Table 1: Operation counts for the full extension field arithmetic

Applying the technique of lazy reduction [4] and Karatsuba algorithm, a detailed description of the finite field arithmetic in $\mathbb{F}_{p^{13}}$ was given in [17]. In Table 1, we summarize the associated operation counts. We now provide detailed operation counts of the pairing computation on BW13-310 by using our algorithms. From (19), the computation of $N_{-z,Q}(P)$, $D_{-z,Q}(P)$,$N_{-z,Q}(\hat{\phi}(P))$, $D_{-z,Q}(\hat{\phi}(P))$ requires executing 5 SQPL, 2 SADD and 1 SDBLADD. Thus, the total number of operations in the Miller loop is

$$\begin{aligned}
ML = {} & \underbrace{2\tilde{\mathbf{a}}}_{\text{Eq.(20)}} + \underbrace{12\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 13\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 28\tilde{\mathbf{a}}}_{\text{the first QPL}} + \\
& \underbrace{4(12\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 15\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 28\tilde{\mathbf{a}})}_{\text{the last 4 QPL}} + \\
& \underbrace{2(12\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + 39\mathbf{m} + 3\tilde{\mathbf{r}} + 15\tilde{\mathbf{a}})}_{\text{2 SADD}} + \\
& \underbrace{13\tilde{\mathbf{m}} + 8\tilde{\mathbf{m}}_u + 10\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 39\mathbf{m} + 26\tilde{\mathbf{a}}}_{\text{1 SDBLADD}} + \\
& \underbrace{\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + 6\tilde{\mathbf{m}} + 13\mathbf{m} + 5\tilde{\mathbf{f}} + 3\tilde{\mathbf{a}}}_{L_1 \text{ and } L_2} \\
& = \tilde{\mathbf{e}} + \tilde{\mathbf{i}} + 103\tilde{\mathbf{m}} + 48\tilde{\mathbf{m}}_u + 89\tilde{\mathbf{s}} + 11\tilde{\mathbf{s}}_u + 42\tilde{\mathbf{r}} + 260\mathbf{m} + 65\mathbf{m}_u + 5\tilde{\mathbf{f}} + 201\tilde{\mathbf{a}} \\
& = \mathbf{i} + 393\mathbf{m} + 10506\mathbf{m}_u + 7326\mathbf{s}_u + 3277\mathbf{r} + 131128\mathbf{a}.
\end{aligned}$$

It can be seen from Section 3.2 that the total number of operations in the final exponentiation is

$$\begin{aligned}
FE = {} & (\tilde{\mathbf{i}} + 3\tilde{\mathbf{m}} + 2\tilde{\mathbf{f}}) + (4\tilde{\mathbf{e}} + 4\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}}) + (10\tilde{\mathbf{e}} + \tilde{\mathbf{m}}) + (4\tilde{\mathbf{e}} + 7\tilde{\mathbf{m}}) \\
& + (\tilde{\mathbf{i}}_c + 9\tilde{\mathbf{m}} + 4\tilde{\mathbf{f}}) + (9\tilde{\mathbf{e}} + 5\tilde{\mathbf{m}} + 4\tilde{\mathbf{f}}) \\
= {} & 27\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + \tilde{\mathbf{i}}_c + 29\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}} + 10\tilde{\mathbf{f}} \\
= {} & \mathbf{i} + 253\mathbf{m} + 7801\mathbf{m}_u + 19866\mathbf{s}_u + 5448\mathbf{r} + 192605\mathbf{a}.
\end{aligned}$$

In Table 2, we compare the operation counts of ML and FE on BW13-310 to the previous works available in the literature. It should be noted that the estimation in [31] assumes that $\tilde{\mathbf{m}} \approx 59\mathbf{m}$, while in [23] assumes that $\tilde{\mathbf{m}} \approx 66\mathbf{m}$. Clearly, our algorithms require fewer computational cost as compared to the previous works.

Work	Phase	Operation Counts
Guillevic [31]	ML	$\approx 2\tilde{\mathbf{i}} + 29919\mathbf{m}$
	FE	$-$
Dai *et al.* [17]	ML	$2\mathbf{i}+518\mathbf{m}+15798\mathbf{m}_u+10758\mathbf{s}_u+4747\mathbf{r}+195187\mathbf{a}$
	FE	$\mathbf{i}+325\mathbf{m}+7801\mathbf{m}_u+19932\mathbf{s}_u+5461\mathbf{r}+193048\mathbf{a}$
Fouotsa *et al.* [23]	ML	$\approx 2\mathbf{i} + 22925\mathbf{m}$
	FE	$-$
This work	ML	$\mathbf{i}+393\mathbf{m}+10506\mathbf{m}_u+7326\mathbf{s}_u+3277\mathbf{r}+131128\mathbf{a}$
	FE	$\mathbf{i}+253\mathbf{m}+7801\mathbf{m}_u+19866\mathbf{s}_u+5448\mathbf{r}+192605\mathbf{a}.$

Table 2: Comparison of operation counts for pairing computation on BW13-310 with the previous works available in the literature

4 Pairings Products Computation on BW13-310

The evaluation of the products of pairings is often required in many pairing-based protocols. Efficient algorithms for computing such products were proposed in [29,46,51] by sharing an amount of full extension field squarings and the costly final exponentiation step. In the case of BW13-310, the products of n-pairings can be expressed as

$$\begin{aligned}\prod_{i=1}^{n} e(P_i, Q_i) &= \Big(\big(\prod_{i=1}^{n} f_{-z,Q_i}(P_i)\big)^{-z+p} \cdot \big(\prod_{i=1}^{n} f_{-z,Q_i}(\hat{\phi}(P_i))\big)^{p} \cdot \prod_{i=1}^{n} \big(y_{P_i} - y_{Q_i}^{p^2}\big)\Big)^{(p^{13}-1)/r} \\ &= (L_{n,1}/L_{n,2})^{(p^{13}-1)/r}.\end{aligned}$$

By Eqs. (24) and (25), the values of $L_{n,1}$ and $L_{n,2}$ are given by

$$\begin{aligned}L_{n,1} &= \Big(\frac{\prod_{i=1}^{n} N_{-z,Q_i}(P_i)}{\prod_{i=1}^{n} D_{-z,Q_i}(P_i) \cdot \prod_{i=1}^{n}(x_{P_i} - \omega \cdot x_{Q_i}^{p})}\Big)^{p-z} \cdot \Big(\prod_{i=1}^{n} N_{-z,Q_i}(\hat{\phi}(P_i))\Big)^{p} \cdot \prod_{i=1}^{n} (y_{P_i} - y_{Q_i}^{p^2}), \\ L_{n,2} &= \Big(\prod_{i=1}^{n} D_{-z,Q_i}(\hat{\phi}(P_i)) \cdot \prod_{i=1}^{n}(\tilde{x}_{P_i} - \omega \cdot x_{Q_i}^{p})\Big)^{p},\end{aligned} \tag{29}$$

where $\tilde{x}_{P_i}$ represents the x-coordinate of $\hat{\phi}(P_i)$. Clearly, the most costly operations for computing $L_{n,1}$ and $L_{n,2}$ take place in the evaluations of $\prod_{i=1}^{n} N_{-z,Q_i}(P_i)$, $\prod_{i=1}^{n} D_{-z,Q_i}(P_i)$, $\prod_{i=1}^{n} N_{-z,Q_i}(\hat{\phi}(P_i))$ and $\prod_{i=1}^{n} D_{-z,Q_i}(\hat{\phi}(P_i))$. To start, we need to obtain the following four initial values:

$$\begin{aligned}
&\prod_{i=1}^{n} N_{1,Q_i}(P_i) = \prod_{i=1}^{n}(x_{P_i} - x_{Q_i}), && \prod_{i=1}^{n} D_{1,Q_i}(P_i) = 1,\\
&\prod_{i=1}^{n} N_{1,Q_i}(\hat{\phi}(P_i)) = \prod_{i=1}^{n}(\tilde{x}_{P_i} - x_{Q_i}), && \prod_{i=1}^{n} D_{1,Q_i}(\hat{\phi}(P_i)) = 1.
\end{aligned} \tag{30}$$

Given an integer m, we denote T_i by $[m]Q_i$ for each point Q_i. Then, the following relations are easily derived from Section 3.1:

$$\text{nSADD}\begin{cases} \displaystyle\prod_{i=1}^{n} N_{m+1,Q_i}(x,y) = \prod_{i=1}^{n} N_{m,Q_i}(x,y) \prod_{i=1}^{n} L_{T_i,a,1}(x,y),\\ \displaystyle\prod_{i=1}^{n} D_{m+1,Q_i}(x,y) = \prod_{i=1}^{n} D_{m,Q_i}(x,y) \prod_{i=1}^{n} L_{T_i,a,2}(x,y). \end{cases}$$

$$\text{nSDBL}\begin{cases} \displaystyle\prod_{i=1}^{n} N_{2m+1,Q_i}(x,y) = \Big(\prod_{i=1}^{n} N_{m,Q_i}(x,y)\Big)^2 \prod_{i=1}^{n} L_{T_i,d,1}(x,y),\\ \displaystyle\prod_{i=1}^{n} D_{2m+1,Q_i}(x,y) = \Big(\prod_{i=1}^{n} D_{m,Q_i}(x,y)\Big)^2 \prod_{i=1}^{n} L_{T_i,d,2}(x,y). \end{cases}$$

$$\text{nSQPL}\begin{cases} \displaystyle\prod_{i=1}^{n} N_{4m,Q_i}(x,y) = \Big(\prod_{i=1}^{n} N_{m,Q_i}(x,y)\Big)^4 \prod_{i=1}^{n} L_{T_i,q,1}(x,y),\\ \displaystyle\prod_{i=1}^{n} D_{4m,Q_i}(x,y) = \Big(\prod_{i=1}^{n} N_{m,Q_i}(x,y)\Big)^4 \Big(\prod_{i=1}^{n} L_{T_i,q,2}(x,y)\Big)^2. \end{cases}$$

Based on the analysis of Section 3.1, we can deduce that the costs of the nSADD, nSDBLADD and nSQPL steps are

- nSADD: $n(12\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + 39\mathbf{m} + 3\tilde{\mathbf{r}} + 15\tilde{\mathbf{a}})$.
- nSDBLADD:

$$\begin{aligned}
&\underbrace{n(2\tilde{\mathbf{m}} + \tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + \tilde{\mathbf{r}} + 7\tilde{\mathbf{a}})}_{\text{point doublings}} + \underbrace{n(6\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + \tilde{\mathbf{r}} + 8\tilde{\mathbf{a}})}_{\text{point additions}}\\
&+ \underbrace{n(\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 4\tilde{\mathbf{r}} + 39\mathbf{m} + 11\tilde{\mathbf{a}})}_{L_{T_i,d,1} \text{ and } L_{T_i,d,2}} + \underbrace{4n\tilde{\mathbf{m}} + 4\tilde{\mathbf{s}}}_{\text{the final step}}\\
&= 4\tilde{\mathbf{s}} + n(13\tilde{\mathbf{m}} + 8\tilde{\mathbf{m}}_u + 6\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 39\mathbf{m} + 26\tilde{\mathbf{a}}.
\end{aligned}$$

- nSQPL:

$$\begin{aligned}
&\underbrace{n(4\tilde{\mathbf{m}} + 2\tilde{\mathbf{m}}_u + 6\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 2\tilde{\mathbf{r}} + 14\tilde{\mathbf{a}})}_{\text{point quadruplings}} +\\
&\underbrace{n(4\tilde{\mathbf{m}} + 4\tilde{\mathbf{m}}_u + \tilde{\mathbf{s}} + 4\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 14\tilde{\mathbf{a}})}_{L_{T_i,q,1} \text{ and } L_{T_i,q,2}} + \underbrace{4n\tilde{\mathbf{m}} + 8\tilde{\mathbf{s}}}_{\text{the final step}}\\
&= 8\tilde{\mathbf{s}} + n(12\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 7\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 28\tilde{\mathbf{a}}).
\end{aligned}$$

Analogous to the single pairing computation, two full extension field squarings can be saved at the first nSQPL step, and we can obtain the terms $\prod_{i=1}^{n} N_{-z,Q_i}(P_i)$, $\prod_{i=1}^{n} D_{-z,Q_i}(P_i)$, $\prod_{i=1}^{n} N_{-z,Q_i}(\hat{\phi}(P_i))$ and $\prod_{i=1}^{n} D_{-z,Q_i}(\hat{\phi}(P_i))$ by executing 5 nSQPL, 2 nSADD and 1 nSDBLADD. On this basis, we continue to calculate $L_{n,1}$ and $L_{n,2}$. Since the point P_i for each i is defined over $\mathbb{F}_p$, we have

$$\begin{aligned}
\prod_{i=1}^{n}(x_{P_i} - \omega \cdot x_{Q_i}^{p}) &= \big(\prod_{i=1}^{n}(x_{P_i} - \omega \cdot x_{Q_i})\big)^{p}, \\
\prod_{i=1}^{n}(\tilde{x}_{P_i} - \omega \cdot x_{Q_i}^{p}) &= (\prod_{i=1}^{n}(\tilde{x}_{P_i} - \omega \cdot x_{Q_i}))^{p}, \\
\prod_{i=1}^{n}\big(y_{P_i} - y_{Q_i}^{p^2}\big) &= \big(\prod_{i=1}^{n}\big(y_{P_i} - y_{Q_i}\big)\big)^{p^2}.
\end{aligned}$$

The above computation requires $3(n-1)\tilde{\mathbf{m}} + 13n\mathbf{m} + 3\tilde{\mathbf{f}} + 3n\tilde{\mathbf{a}}$. By the form of Eq. (29), it is straightforward to see that the cost of computing $L_{n,1}$ and $L_{n,2}$ is

$$\begin{aligned}
&(3(n-1)\tilde{\mathbf{m}} + 13n\mathbf{m} + 3\tilde{\mathbf{f}} + 3n\tilde{\mathbf{a}}) + (\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + 6\tilde{\mathbf{m}} + 2\tilde{\mathbf{f}}) \\
=&\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + 3(n+1)\tilde{\mathbf{m}} + 13n\mathbf{m} + 5\tilde{\mathbf{f}} + 3n\tilde{\mathbf{a}}.
\end{aligned}$$

Based on the above analysis, the cost of the Miller loop for computing n-pairings products on BW13-310 is

$$\begin{aligned}
nML =& \underbrace{2(n-1)\tilde{\mathbf{m}} + 2n\tilde{\mathbf{a}}}_{\text{Eq. (30)}} + \underbrace{6\tilde{\mathbf{s}} + n(12\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 7\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 28\tilde{\mathbf{a}})}_{\text{the first nQPL}} + \\
& \underbrace{32\tilde{\mathbf{s}} + 4n(12\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 7\tilde{\mathbf{s}} + 2\tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 26\mathbf{m} + 13\mathbf{m}_u + 28\tilde{\mathbf{a}})}_{\text{the last 4 nQPL}} + \\
& \underbrace{2n(12\tilde{\mathbf{m}} + 5\tilde{\mathbf{m}}_u + 3\tilde{\mathbf{s}} + 39\mathbf{m} + 3\tilde{\mathbf{r}} + 15\tilde{\mathbf{a}})}_{\text{2 nSADD}} + \\
& \underbrace{4\tilde{\mathbf{s}} + n(13\tilde{\mathbf{m}} + 8\tilde{\mathbf{m}}_u + 6\tilde{\mathbf{s}} + \tilde{\mathbf{s}}_u + 6\tilde{\mathbf{r}} + 39\mathbf{m} + 26\tilde{\mathbf{a}})}_{\text{1 nSDBLADD}} + \\
& \underbrace{\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + 3(n+1)\tilde{\mathbf{m}} + 13n\mathbf{m} + 5\tilde{\mathbf{f}} + 3n\tilde{\mathbf{a}}}_{L_{n,1} \text{ and } L_{n,2}} \\
=&\tilde{\mathbf{e}} + \tilde{\mathbf{i}} + (102n+1)\tilde{\mathbf{m}} + 48n\tilde{\mathbf{m}}_u + (47n+42)\tilde{\mathbf{s}} + 11n\tilde{\mathbf{s}}_u + 42n\tilde{\mathbf{r}} + 260n\mathbf{m} \\
& + 65n\mathbf{m}_u + 5\tilde{\mathbf{f}} + 201n\tilde{\mathbf{a}}. \\
=&\mathbf{i} + (260n+133)\mathbf{m} + (9965n+541)\mathbf{m}_u + (3828n+3498)\mathbf{s}_u + (2483n+794)\mathbf{r} \\
& + (103607n+27521)\mathbf{a}.
\end{aligned}$$

In summary, the total number of operations required for computing n-pairings products on BW13-310 is

$$\begin{aligned}
nML + FE =& 2\mathbf{i} + (260n+386)\mathbf{m} + (9965n+8342)\mathbf{m}_u + (3828n+23364)\mathbf{s}_u \\
& + (2483n+6242)\mathbf{r} + (103607n+220126)\mathbf{a}.
\end{aligned}$$

5 Exponentiation in Pairing Groups

Exponentiation in three pairing groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ also plays a vital role in the implementation of pairing-based protocols. In this section, we discuss how to efficiently perform the operation on BW13-310. The best-known method for computing $[n]P$ with $P \in \mathbb{G}_1$ and $n \in \mathbb{Z}_r$ was introduced by Gallant, Lambert, Vanstone [28], which is called the GLV method.The basic idea of this method is as follows. If the target curve has an efficiently computable endomorphism ϕ such that $\phi(P) = [\lambda]P$ for some integer λ, then the scalar n can be decomposed as n_0, n_1 such that $n \equiv n_0 + n_1 \cdot \lambda \bmod r$, where $|n_0|, |n_0| \approx \sqrt{r}$. Thus, the computation of $[n]P$ can be replaced by the multi-exponentiation $[n_0]P + [n_1]\phi(P)$. In summary, this method can halve the number of point doublings. In addition, recoding n_0 and n_1 with the w-width non-adjacent form (w-NAF) can reduce the number of point additions. In the RELIC cryptographic toolkit [1], this method can be implemented automatically once the associated curve parameters are given. Therefore, we only investigate how to perform exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$ on this curve.

5.1 Exponentiation in $\mathbb{G}_2$

For exponentiation in $\mathbb{G}_2$ on curves admitting a twist, such as BN and BLS12 families, GLS method [26] breaks a random exponent $n \in \mathbb{Z}_r$ into $\varphi(k)$ mini-exponents $n_0, n_1, \cdots, n_{\varphi(k)-1}$ such that the bit size of the maximum of $|n_i|$ is about $\frac{1}{\varphi(k)} \log r$. In general, GLS method largely reduces the number of iterations required for computing $[n]Q$ with $Q \in \mathbb{G}_2$.

It is possible to build a higher-dimensional decomposition by combining GLV and GLS methods on some certain curves. This idea was initially proposed by Longa and Sica [40] to obtain a four dimensional decomposition on certain curves, and subsequently applied into different scenarios [14,22]. On BW13-310, the orders of ϕ and π are respectively 3 and k in $\mathrm{End}_{\mathbb{F}_{p^k}}(E)$ with $\gcd(3, k) = 1$, where $\mathrm{End}_{\mathbb{F}_{p^k}}(E)$ denotes the endomorphism ring of E over $\mathbb{F}_{p^k}$. It indicates that $\psi = \phi \circ \pi$ satisfies $\Phi_{2\varphi(k)}(\psi) = 0$, so a random exponent n can be decomposed into $2\varphi(k)$ mini-exponents, where the bit size of the maximum of $|n_i|$ is about $\frac{1}{2\varphi(k)} \log r$. It should be noted that the $2\varphi(k)$-dimensional decomposition takes advantage of the fact that $\psi(Q) = [-z]Q$. More specifically, since $\log r \approx 24 \log|z|$, the exponent n can be written in the basis of $|z|$ as

$$n = n_0 + n_1 \cdot |z| + \cdots + n_{23} \cdot |z|^{23},$$

where $\log|n_i| < \log|z| \approx \frac{1}{24} \log r$. Thus, the computation of $[n]Q$ with $Q \in \mathbb{G}_2$ can be accomplished as

$$[n]Q = [n_0]Q + n_1\psi(Q) + \cdots + [n_{23}]\psi^{23}(Q).$$

By the form of the endomorphism ψ, we have

$$\psi^i(Q) = (\omega^i \cdot x_Q^{p^i}, y_Q^{p^i})$$

for $i = 0, 1, \cdots, 23$, and thus the cost of computing $\psi^i(Q)$ is negligible. The procedure of performing exponentiation in $\mathbb{G}_2$ on BW13-310 is summarized in Alg. 2.

Remark 1. We conclude that if the orders of ϕ and π are coprime in $\mathrm{End}_{\mathbb{F}_{p^k}}(E)$, then there exists a $2\varphi(k)$-dimensional decomposition in $\mathbb{G}_2$. However, the orders of ϕ and π are respectively d and k in $\mathrm{End}_{\mathbb{F}_{p^k}}(E)$ with $d \mid k$ ($d = 3$ or 4) on many mainstream pairing-friendly curves, such as BN, BLS12, KSS16 and KSS18 families. It means that $\psi = \phi \circ \pi$ satisfies $\Phi_{\varphi(k)}(\psi) = 0$. Thus, the new technique is not suitable for the above mentioned curves.

Algorithm 2 Exponentiation in $\mathbb{G}_2$ on BW13-310

```
Input: a random positive integer $n \in \mathbb{Z}_r$, a random point $Q \in \mathbb{G}_2$
Output: $[n]Q$
$Q_0 \leftarrow Q$
for $i = 1$ to 23 do
    $Q_i \leftarrow \psi(Q_{i-1})$
end for
Compute $[j]Q_i$ for $j \in \{1, 3, 5, \cdots, 2^w - 1\}$
Decompose $n$ into $(n_0, n_1, \cdots, n_{23})$ with $n = \sum_{i=0}^{23} n_i \cdot |z|^i$
Recode $n_i = \sum_{j=0}^{t-1} n_{i,j} 2^j$ in $w$-NAF
$R \leftarrow \mathcal{O}$
for $j = t - 1$ down to 0 do
    $R \leftarrow [2]R$
    for $i = 0$ to 23 do
        if $n_{i,j} > 0$
            $R \leftarrow R + [n_{i,j}]Q_i$
        else
            $R \leftarrow R - [n_{i,j}]Q_i$
        end if
    end for
end for
return $R$
```

5.2 Exponentiation in $\mathbb{G}_T$

Since there is not an efficient computable endomorphism in $\mathbb{F}_{p^k}$ corresponding to the GLV endomorphism in $E(\mathbb{F}_{p^k})$, the exponentiation in $\mathbb{G}_T$ is slightly different to that in $\mathbb{G}_2$. In other words, in the case of exponentiation in $\mathbb{G}_T$, a given exponent n can be only decomposed into $\varphi(k)$ multi-exponents by using the Frobenius endomorphism, rather than $2\varphi(k)$. We now follow the same recipe described by Galbraith and Scott [27] to decompose n the as $n_0, n_1, \cdots, n_{\varphi(k)-1}$ such that

$$n \equiv (n_0 + n_1 \cdot p + \cdots + n_{\varphi(k)-1} p^{\varphi(k)-1}) \bmod r$$

and $\max\{|n_0|, |n_1| \cdots, |n_{\varphi(k)-1}|\} \approx r^{1/\varphi(k)}$. To this aim, we first define a modular lattice $\mathcal{L}$ as

$$\mathcal{L} = \{(z_0, z_1, \cdots, z_{11}) | z_0 + z_1 \cdot p + \cdots + z_{11} \cdot p^{11} \equiv 0 \bmod r\}.$$

Clearly, a basis $\mathbf{B}^* = (\mathbf{b}_0^*, \mathbf{b}_1^*, \cdots, \mathbf{b}_{11}^*)$ of $\mathcal{L}$ is naturally selected as

$$\begin{aligned} \mathbf{b}_0^* &= (r, 0, 0, \cdots, 0), \\ \mathbf{b}_1^* &= (p, -1, 0 \cdots, 0), \\ &\vdots \\ \mathbf{b}_{11}^* &= (p, 0, 0 \cdots, -1). \end{aligned}$$

Inputting the basis $\mathbf{B}^*$ into the LLL algorithm [38], we obtain a LLL-reduced basis $\mathbf{B} = (\mathbf{b}_0, \mathbf{b}_1, \cdots, \mathbf{b}_{11})$ as

$$\begin{aligned} \mathbf{b}_0 &= (z^2, -z, 1, 0, 0, \cdots, 0), \\ \mathbf{b}_1 &= (0, z^2, -z, 1, 0, \cdots, 0), \\ &\vdots \\ \mathbf{b}_9 &= (0, 0, \cdots, 0, z^2, -z, 1), \\ \mathbf{b}_{10} &= (1, \cdots, 1, -z^2 + 1, z + 1), \\ \mathbf{b}_{11} &= (-z - 1, -z, \cdots, -z, -z^2 + z). \end{aligned}$$

Since $(r, 0, 0, \cdots, 0) \in \mathcal{L}$, there exists a unique solution $(y_0, y_1, \cdots, y_{11}) \in \mathbb{Z}^{12}$ such that

$$(r, 0, 0, \cdots, 0) = y_0 \mathbf{b}_0 + y_1 \mathbf{b}_1 + \cdots + y_{11} \mathbf{b}_{11} \tag{31}$$

Multiplying n/r on the both side of (31), it produces that

$$(n, 0, 0, \cdots, 0) = \alpha_0 \mathbf{b}_0 + \alpha_1 \mathbf{b}_1 + \cdots + \alpha_{11} \mathbf{b}_{11}, \tag{32}$$

where $\alpha_i = y_i \cdot n/r$. We define

$$\mathbf{n} = (n_0, n_1, \cdots, n_{11}) = (n, 0, 0, \cdots, 0) - \lfloor \alpha_0 \rceil \mathbf{b}_0 - \lfloor \alpha_1 \rceil \mathbf{b}_1 - \cdots - \lfloor \alpha_{11} \rceil \mathbf{b}_{11}. \tag{33}$$

Since $\lfloor \alpha_i \rceil \in \mathbb{Z}$ for each i, it is clear that $\mathbf{n} \in n + \mathcal{L}$. Moreover, combining Eqs.(32) and (33), we have

$$(n_0, n_1, \cdots, n_{11}) = (\alpha_0 - \lfloor \alpha_0 \rceil) \mathbf{b}_0 + (\alpha_1 - \lfloor \alpha_1 \rceil) \mathbf{b}_1 + \cdots + (\alpha_{11} - \lfloor \alpha_{11} \rceil) \mathbf{b}_{11}.$$

Since the selected seed z is negative and $|\alpha_0 - \lfloor \alpha_0 \rceil| \leq 1/2$, we immediately get that

$$\|\mathbf{n}\|_\infty \leq (z^2 - 2z + 2)/2 \approx r^{1/\varphi(k)}. \tag{34}$$

For the mainstream pairing-friendly curves, such as BN, BLS12, KSS16 and KSS18 families, the embedding degrees k are always even. This allows inversion in $\mathbb{G}_T$ to be computed for almost free by using a simple conjugation over $\mathbb{F}_{p^{k/2}}$. In other words, non-positive exponent decomposition will not bring extra overhead for exponentiation in $\mathbb{G}_T$ on these curves. However, the picture is different on BW13-310 as it has an odd prime embedding degree. It pays a penalty for expensive cost inversion in $\mathbb{G}_T$. In order to avoid this operation when performing exponentiation in $\mathbb{G}_T$, we expect multi-exponents are all-positive.

Proposition 1. *Let the lattice $\mathcal{L}$ and the LLL-reduced basis $\boldsymbol{B} = (\mathbf{b}_0, \mathbf{b}_1, \cdots, \mathbf{b}_{11})$ of $\mathcal{L}$ be constructed as above. For any integer $n \in \mathbb{Z}$, there exists a vector $\mathbf{n}' = (n'_0, n'_1, \cdots, n'_{11}) \in n + \mathcal{L}$ such that the tuples of $\mathbf{n}'$ are all-positive and $\|\mathbf{n}'\|_\infty \leq 3(z^2 - 2z + 2)/2$.*

Proof. We first decompose the scalar n into the vector $\mathbf{n}$ as in (33). Then, we define $\mathbf{c} = \mathbf{b}_0 + \mathbf{b}_1 + \cdots + \mathbf{b}_9 - \mathbf{b}_{10} - \mathbf{b}_{11}$ and $\mathbf{n}' = \mathbf{c} + \mathbf{n}$. It is obviously that $\mathbf{n}' \in n + \mathcal{L}$. Moreover, by the definition of $\mathbf{b}_i$ for each i, it is straightforward to see that

$$\min\{c_0, c_1, \cdots, c_{\varphi(k)-1}\} \geq z^2 - 1, \max\{c_0, c_1, \cdots, c_{\varphi(k)-1}\} \leq z^2 - 2z + 2, \tag{35}$$

where c_i is denoted as the i-th tuple of $\mathbf{c}$. Combining Eq.s (34) and (35) together, the proof is immediate.

The all-positive decomposition described in Proposition 1 also leads to around 1 bit increase for the bound of the size of the mini-exponents. Considering the expensive cost of inversion in $\mathbb{G}_T$, this trade is absolutely worthwhile. In addition, one should be noted that when performing the small exponentiations by n'_i for $i = 0, 1, \cdots, 11$, the NAF expression is not applicable. The procedure of exponentiation in $\mathbb{G}_T$ on BW13-310 is presented in Alg. 3.

6 Implementation Results

In this section, we present our implementation results of the pairing computation and common building blocks on BW13-310 within the RELIC cryptographic toolkit. For the pairing computation and the group exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$ we use the algorithms proposed in this paper. For hashing to $\mathbb{G}_1$ and $\mathbb{G}_2$, the group membership testings and the group exponentiation in $\mathbb{G}_1$, we exploit state-of-the-art techniques. In detail,

Algorithm 3 Exponentiation in $\mathbb{G}_T$ on BW13-310

```
Input: a random positive integer $n \in \mathbb{Z}_r$, a random element $f \in \mathbb{G}_T$
Output: $f^n$
$f_0 \leftarrow f$
for $i = 1$ to $11$ do
    $f_i \leftarrow f_{i-1}^p$
end for
Compute $f_i^j$ for $j \in \{1, 3, 5, \cdots, 2^w - 1\}$
Decompose $n$ into all-positive mini-exponents $(n_0, \cdots, n_{11})$ by using Proposition 1
Recode $n_i = \sum_{j=0}^{t-1} n_{i,j} 2^j$ such that $n_{i,j} \in \{1, 3, 5, \cdots, 2^w - 1\}$
$g \leftarrow 1$
for $j = t - 1$ down to $0$ do
    $g \leftarrow g^2$
    for $i = 0$ to $11$ do
        if $n_{i,j} > 0$
            $g \leftarrow g \cdot f_i^{n_{i,j}}$
        end if
    end for
end for
return $g$
```

- The function $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ is implemented by using the Shallue-van de Woestijne (SVW) map [48], followed by a cofactor multiplication. The SVW map aims to efficiently hash a binary string to a random point $R_1 \in E(\mathbb{F}_p)$ in constant time, and the cofactor multiplication forces R_1 into the target group $\mathbb{G}_1$. RELIC provides dedicated implementation of this map in the *ep_map_from_field*() function in the file */src/ep/relic_ep_map.c*. The procedure of clearing cofactor can be done at a cost of one multiplication by $z^2 - z + 1$ [20].
- The function $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$ is implemented by using the method proposed in [16]. Likewise, it is split into two phases: hashing a binary string to a random point $R_2 \in E(\mathbb{F}_{p^{13}})$, followed by mapping R_2 to $\mathbb{G}_2$. The computational cost of hashing to $\mathbb{G}_2$ largely comes from the second phase, which requires approximately 26 scalar multiplications by z, 13 point doublings and 41 point additions in $E(\mathbb{F}_{p^{13}})$.
- The best-known algorithms for the $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ membership testings on BW13-310 are proposed in [15], which require approximately 12 scalar multiplications by z in $E(\mathbb{F}_p)$, 1 scalar multiplications by z in $E(\mathbb{F}_{p^{13}})$ and 2 exponentiations by z in $\mathbb{F}_{p^{13}}$, respectively.

In order to evaluate strengths and weaknesses of BW13-310 in real-world pairing based cryptographic protocols, we present a performance comparison between BW13-310 and other 128-bit secure pairing-friendly curves, including BN446, BLS12-446 and BLS24-315. Specially, BN446 and BLS12-446 are well known for fast pairing computations, while BLS24-315 is another interesting curve with fast exponentiation in $\mathbb{G}_1$. All of these curves are defined by an equation of the form $y^2 = x^3 + b$ for some $b \in \mathbb{F}_p^*$, and the related parameters are summarized in Table 3. RELIC provides high speed implementations of pairing computations and the required auxiliary building blocks on BN446, BLS12-446 and BLS24-315. We have integrated our codes in this library to allow a direct performance comparison across different curves. The source code is available at https://github.com/eccdaiy39/BW13-P310. Our benchmark results are presented in **Figs**. 1-4. Timings are measured on an Intel Core i9-12900K processor running at @3.2GHz with TurboBoost and hyper-threading features disabled, averaged over 10^4 executions. For group exponentiations $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on each curve, window widths w are set as 4, 1 and 1, respectively.

	p	r	z, b
BN446	$36z^4+36z^3+24z^2+6z+1$	$36z^4+36z^3+18z^2+6z+1$	$2^{110}+2^{36}+1, 257$
BLS12-446	$(z-1)^2(z^4-z^2+1)/3+z$	z^4-z^2+1	$-2^{74}+2^{73}-2^{63}-2^{57}-2^{50}-2^{17}-1, 1$
BW13-310	$\frac{1}{3}(z+1)^2(z^{26}-z^{13}+1)-z^{27}$	$\Phi_{78}(z)$	$-(2^{11}+2^7+2^5+2^4), -17$
BLS24-315	$(z-1)^2(z^8-z^4+1)/3+z$	z^8-z^4+1	$-2^{32}+2^{30}+2^{21}+2^{20}+1, 1$

Table 3: Important parameters of BN446, BLS12-446, BW13-310 and BLS24-315.

- Compared to BN446 and BLS12-446, BW13-310 is about $109.1\% - 227.3\%$, $100\% - 192.6\%$, $24.5\% - 108.5\%$ and $68.2\% - 145.5\%$ faster in terms of hashing to $\mathbb{G}_1$, exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, and membership testing for $\mathbb{G}_T$, respectively. In essence, as to operations related to $\mathbb{G}_1$, BW13-310 benefits from fast prime field arithmetic. As to operations related to $\mathbb{G}_T$, even though BW13-310 fails to provide fast cyclotomic squaring [30, 34] and decode exponents in the NAF form, it is much more favoured for a small size of full extension field and a large value of $\varphi(k)$, which result in fast full extension field multiplication and high dimensional GLS decomposition.
- More surprisingly, the gap in the performance of single pairing computation between BW13-310 and BN446 (resp. BLS12-446) is only up to 4.9% (resp. 26%). In particular, the computation of the Miller loop on BW13-310 is even up to 48.2% faster than that on BN446. In fact, our results reveal that a few percent efficiency disadvantage of pairing computation on BW13-310 mainly arises from the final exponentiation part. For the computation of the n-pairings products, BW13-310 outperforms BN446, while still slower than BLS12-446. For example, for the 8-pairings products, BW13-310 is about

14.2% faster than that on BN446, while 25.5% slower than that on BLS12-446.

- However, BW13-310 also introduces a significant penalty for hashing to $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_2$. Indeed, the pairing group $\mathbb{G}_2$ on BW13-310 is defined over the full extension field as the lack of twists, while that on BLS12-446 and BN446 lies in a subfield $\mathbb{F}_{p^2}$.
- For BW13-310 and BLS24-315, they provide nearly equal performance in terms of exponentiation in $\mathbb{G}_1$, membership testing for $\mathbb{G}_1$. Moreover, the former has a significant advantage in terms of hashing to $\mathbb{G}_1$, exponentiation in $\mathbb{G}_T$, membership testing for $\mathbb{G}_T$ and pairing computation, while the latter outperforms for hashing to $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_2$.

In summary, our implementation results show that BW13-310 is competitive in scenarios that hashing to $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_2$ are not necessary, or performed by a powerful computational entity.

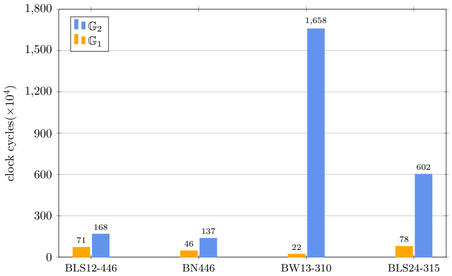

Fig. 1: Hashing to $\mathbb{G}_1$ and $\mathbb{G}_2$.

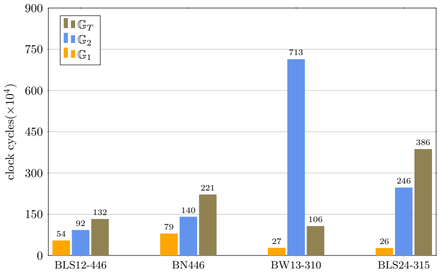

Fig. 2: Group exponentiations

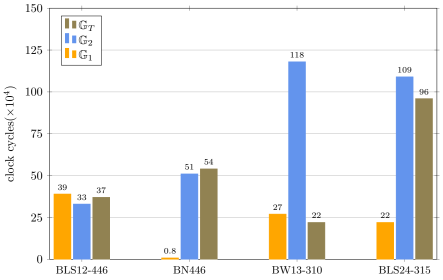

Fig. 3: Group membership testings

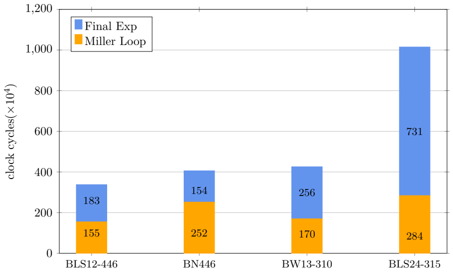

Fig. 4: Single pairing computation

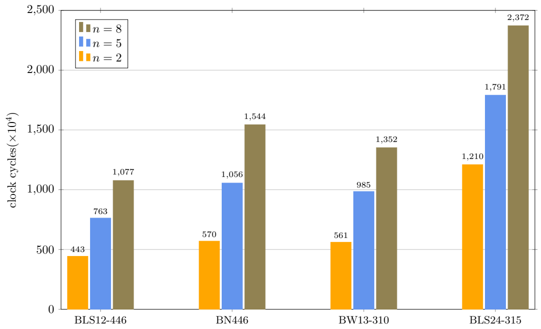

Fig. 5: n-pairings products computation

7 Applications

In this section, we estimate the performance of two pairing-based cryptosystems built on the above mentioned pairing-friendly curves, aiming to explain that BW13-310 is an interesting candidate.

7.1 Unbalanced Chen-Kudla protocol

In [12], Chen and Kudla designed a two-party identity based authenticated key agreement protocol from pairings. In this protocol, each entity is required to perform one exponentiation in $\mathbb{G}_1$ and one pairing computation. In real-world protocols, it is often the case that one entity (Client) is equipped with a resource-constrained device, while the other (Server) is more powerful. To reduce the workload of the client, it is reasonable to shift the time-consuming pairing computation to the server. To this aim, Scott [47] proposed an unbalanced Chen-Kudla protocol (UCK). In this scenario, a system wide public parameter $Q \in \mathbb{G}_2$ is introduced. A trusted authority (TA) posseses a master secret key $s \in \mathbb{Z}_r$. The client and server have secret keys S_A and S_B issued by TA as $S_A = [s]\mathcal{H}_1(ID_A) \in \mathbb{G}_1$ and $S_B = [s]Q \in \mathbb{G}_2$ respectively, where ID_A is the client's identity. Then the protocol runs as follows:

1. The client chooses $r_1, r_2 \leftarrow \mathbb{Z}_r$ at random, calculates $R_1 = [r_1]S_A$ and $R_2 = [r_2]\mathcal{H}_1(ID_A)$ and sends the two points R_1 and R_2 to the server;
2. The server chooses $r_3 \leftarrow \mathbb{Z}_r$ at random, calculates $R_3 = [r_3]R_1$ and $g = e(R_3, Q)$, and send the pairing value g to the client;
3. The client obtains the session key K_A by computing

$$K_A = \left(e(S_A, Q)^{r_1 \cdot r_2} \cdot g\right)^{1/r_1}.$$

   The server obtains the session key K_B by computing

$$K_B = e(R_2 + [r_3]\mathcal{H}_1(ID_A), S_B).$$

4. If the both entities follow this protocol, they would share the same session key
$$K = K_A = K_B = e(S_A, Q)^{(r_2+r_3)}.$$

Since the client can precompute the point $\mathcal{H}_1(ID_A)$ and the pairing value $e(S_A, Q)$, the entity only costs two exponentiations in $\mathbb{G}_1$ and two exponentiations in $\mathbb{G}_T$. Meanwhile, the server costs one hashing to $\mathbb{G}_1$, two exponentiations in $\mathbb{G}_1$ and two pairing computations.

Considering the protocol is designed to minimize the workload of the client, we can see that BW13-310 is well-suited as it provides fast implementation for group exponentiations in both $\mathbb{G}_1$ and $\mathbb{G}_T$. Based on implementation results presented in Figs.1-4, Table 4 shows our cost estimates for each party of the UCK protocol built on different curves. One can see that the UCK protocol built on BW13-310 is about 125.6% and 40.6% faster than that on BN446 and BLS12-446 for the client, respectively.

Protocol\Curve	BLS12-446	BN446	BW13-310	BLS24-315
preco	409	452	448	1093
client	374	600	266	824
server	855	1016	928	2160

Table 4: Timings of the UCK protocol reported in 10^4 clock cycles (extrapolation from Figs.1-4.)

7.2 BLS signature scheme

The Boneh-Lynn-Shacham (BLS) signature is a famous short signature scheme from pairings [8]. In the scheme, the point $g_2 \in \mathbb{G}_2$ is a public parameter and the signer posseses a pair of key $(s, pk = [s]g_2)$, where s is private and pk is public. Then, the scheme works as follows:

1. To sign a message msg, the signer computes $M = \mathcal{H}_1(msg)$, $sig = [s]M$, and sends the pair (msg, sig) to the verifier.
2. To verify the signed message, the verifier computes $M = \mathcal{H}_1(msg)$ and assert that the signature is valid if and only if $sig \in \mathbb{G}_1$ and $e(sig, pk) = e(M, g_2)$.

It should be noted that an attacker can use the point $sig' = sig + R$ to forge a valid signature as $e(sig', pk) = e(sig, pk)$, where R is an random point in the subgroup $r \cdot E(\mathbb{F}_p)$. Thus, it can not be ignored for the verifier to check $sig \in \mathbb{G}_1$. In this setting, the signer costs one hashing to $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_1$, while the verifier costs one hashing to $\mathbb{G}_1$, one subgroup membership testing for $\mathbb{G}_1$ and one product of 2-pairings. From Figs. 1-4, we estimate that the BLS signature scheme built on BW13-310 is about both 1.5× faster than that on BN446 and BLS12-446 for the signer, respectively. Considering that the performance penalty for the verifier is not expensive, this tradoff becomes favorable in the case that the scheme is designed to reduce the workload of the signer.

Protocol\Curve	BLS12-446	BN446	BW13-310	BLS24-315
sign	125	125	49	104
verify	553	616.8	610	1115

Table 5: Timings of the BLS signature scheme reported in 10^4 clock cycles (extrapolation from Figs.1-4.)

8 Conclusion and Future Work

In this work, we presented a detailed study of a 128-bit secure pairing-friendly curve: BW13-310. We first proposed a new formula for computing the optimal pairing on this curve. Specially, we showed that it requires two evaluations at the same Miller function of bit length approximately $\log r/(2\varphi(k))$. On this basis, we proposed a shared Miller loop such that the two function evaluations can share intermediate values as much as possible. In addition, we also described several optimizations for group exponentiations in $\mathbb{G}_2$ and $\mathbb{G}_T$ on this curve. In the case of $\mathbb{G}_2$, we showed that GLV and GLS method can be combined to build a $2\varphi(k)$ dimensional decomposition. In the case of $\mathbb{G}_T$, our optimization eliminates expensive field inversion.

Finally, we presented high speed implementations of pairing computation, hashing (to $\mathbb{G}_1$ and $\mathbb{G}_2$), group exponentiations and membership testings on a 64-bit processor over BW13-310. The technique of lazy reduction was fully utilized to minimize the number of modular reductions. Our results showed that compared to BN446 and BLS12-446, BW13-310 wins out in the performance of hashing to $\mathbb{G}_1$, group exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, and membership testing for $\mathbb{G}_T$. Furthermore, it was very surprising to find that the gap in the performance of single pairing computations between BW13-310 and BN446 (resp. BLS12-446) is only up to 4.9% (resp. 26%). In particular, compared to BN446, BW13-310 even has certain advantages for the computation of pairings products. Our results also reported that the target curve would pay a penalty for hashing to $\mathbb{G}_2$ and the group exponentiation in $\mathbb{G}_2$.

Very recently, Longa [39] further optimized the technique of lazy reduction such that the penalty of “double-precision” operations can be avoided. We note that the new algorithm gets a greater performance boost on prime fields with smaller sizes, potentially helping BW13-310 become more attractive. In addition, a faster SVW map (SwiftEC) was proposed in [11](ASIACRYPT 2022).

The performance comparison across different pairing-friendly curves using these optimized algorithms are left as future work.

Acknowledgment

We would like to thank the anonymous referees for their valuable comments and suggestion. This work is supported by Guangdong Major Project of Basic and Applied Basic Research(No. 2019B030302008) and the National Natural Science Foundation of China(No. 61972428 and 61972429).

References

1. Aranha, D.F., Gouvêa, C.P.L.: Relic is an efficient library for cryptography. https://github.com/relic-toolkit/relic
2. Aranha, D.F., El Housni, Y., Guillevic, A.: A survey of elliptic curves for proof systems. Designs, Codes and Cryptography (Dec 2022)
3. Aranha, D.F., Fuentes-Castañeda, L., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Implementing pairings at the 192-bit security level. In: Abdalla, M., Lange, T. (eds.) Pairing-Based Cryptography – Pairing 2012. pp. 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. pp. 48–68. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
5. Azarderakhsh, R., Fishbein, D., Grewal, G., Hu, S., Jao, D., Longa, P., Verma, R.: Fast software implementations of bilinear pairings. IEEE Transactions on Dependable and Secure Computing **14**(6), 605–619 (2017)
6. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of Cryptology **32**(4), 1298–1336 (2019)
7. Barbulescu, R., Gaudry, P., Kleinjung, T.: The tower number field sieve. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 31–55. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of cryptology **17**(4), 297–319 (2004)
9. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**(3-4), 235–265 (1997), computational algebra and number theory (London, 1993)
10. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. pp. 132–145. Association for Computing Machinery, New York, NY, USA (2004)
11. Chavez-Saab, J., Rodríguez-Henríquez, F., Tibouchi, M.: Swiftec: Shallue-van de woestijne indifferentiable function to elliptic curves: Faster indifferentiable hashing to elliptic curves. In: Advances in Cryptology – ASIACRYPT 2022. p. 63–92. Springer-Verlag, Berlin, Heidelberg (2023)
12. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: 16th IEEE Computer Security Foundations Workshop, 2003. Proceedings. pp. 219–233. IEEE (2003)

13. Clarisse, R., Duquesne, S., Sanders, O.: Curves with fast computations in the first pairing group. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security. pp. 280–298. Springer International Publishing, Cham (2020)
14. Costello, C., Longa, P.: Four$\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the mersenne prime. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 214–235. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
15. Dai, Y., Lin, K., Zhao, C.A., Zhou, Z.: Fast subgroup membership testings for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on pairing-friendly curves. Designs, Codes and Cryptography (may 2023). https://doi.org/10.1007/s10623-023-01223-7, https://doi.org/10.1007/s10623-023-01223-7
16. Dai, Y., Zhang, F., Zhao, C.A.: Fast hashing to $\mathbb{G}_2$ in direct anonymous attestation. Cryptology ePrint Archive, Paper 2022/996 (2022), https://eprint.iacr.org/2022/996
17. Dai, Y., Zhou, Z., Zhang, F., Zhao, C.A.: Software implementation of optimal pairings on elliptic curves with odd prime embedding degrees. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **105**(5), 858–870 (2022)
18. El Housni, Y., Guillevic, A.: Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security (2020)
19. El Housni, Y., Guillevic, A.: Families of SNARK-friendly 2-chains of elliptic curves. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 367–396. Springer International Publishing, Cham (2022)
20. El Housni, Y., Guillevic, A., Piellard, T.: Co-factor clearing and subgroup membership testing on pairing-friendly curves. In: Batina, L., Daemen, J. (eds.) Progress in Cryptology – AFRICACRYPT 2022. pp. 518–536. Springer Nature Switzerland, Cham (2022)
21. El Mrabet, N., Joye, M.: Guide to pairing-based cryptography. Chapman and Hall/CRC (2016)
22. Faz-Hernández, A., Longa, P., Sánchez, A.H.: Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV-GLS curves. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. pp. 1–27. Springer International Publishing, Cham (2014)
23. Fouotsa, E., Guimagang, L.A., Ayissi, R.: x-superoptimal pairings on elliptic curves with odd prime embedding degrees: BW13-P310 and BW19-P286. Applicable Algebra in Engineering, Communication and Computing (2023). https://doi.org/10.1007/s00200-023-00596-5, https://doi.org/10.1007/s00200-023-00596-5
24. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (2010)
25. Galbraith, S.: Mathematics of Public Key Cryptography. Cambridge University Press (2018), https://www.math.auckland.ac.nz/~sgal018/crypto-book/main.pdf, version 2
26. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009. pp. 518–535. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
27. Galbraith, S.D., Scott, M.: Exponentiation in pairing-friendly groups using homomorphisms. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing-Based Cryptography – Pairing 2008. pp. 211–224. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

28. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 190–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
29. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive, Paper 2006/172 (2006), https://eprint.iacr.org/2006/172, https://eprint.iacr.org/2006/172
30. Granger, R., Scott, M.: Faster squaring in the cyclotomic subgroup of sixth degree extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography – PKC 2010. pp. 209–223. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
31. Guillevic, A.: A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography – PKC 2020. pp. 535–564. Springer International Publishing, Cham (2020)
32. Guillevic, A., Masson, S., Thomé, E.: Cocks-pinch curves of embedding degrees five to eight and optimal ate pairing computation. Designs, Codes and Cryptography **88**(6), 1047–1081 (2020)
33. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE Transactions on Information Theory **52**(10), 4595–4602 (2006)
34. Karabina, K.: Squaring in cyclotomic subgroups. Mathematics of Computation **82**(281), 555–579 (2012)
35. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 543–571. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
36. Kim, T., Jeong, J.: Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In: Fehr, S. (ed.) Public-Key Cryptography – PKC 2017. pp. 388–408. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)
37. Koshelev, D.: Subgroup membership testing on elliptic curves via the tate pairing. Journal of Cryptographic Engineering (Sep 2022)
38. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4) (1982)
39. Longa, P.: Efficient algorithms for large prime characteristic fields and their application to bilinear pairings. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(3), 445–472 (2023). https://doi.org/10.46586/tches.v2023.i3.445-472, https://tches.iacr.org/index.php/TCHES/article/view/10970
40. Longa, P., Sica, F.: Four-dimensional Gallant-Lambert-Vanstone scalar multiplication. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. pp. 718–739. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
41. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology **17**(4), 235–261 (2004)
42. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. Mathematics of computation **48**(177), 243–264 (1987)
43. Pollard, J.M.: A monte carlo method for factorization. Bit Numerical Mathematics **15**(3), 331–334 (1975)
44. Schirokauer, O.: Discrete logarithms and local units. Philosophical Transactions: Physical Sciences and Engineering **345**(1676), 409–423 (1993)
45. Scott, M.: Faster pairings using an elliptic curve with an efficient endomorphism. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) Progress in Cryptology

- INDOCRYPT 2005. pp. 258–269. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

46. Scott, M.: On the efficient implementation of pairing-based protocols. In: Chen, L. (ed.) Cryptography and Coding. pp. 296–308. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
47. Scott, M.: Unbalancing pairing-based key exchange protocols. IACR Cryptol. ePrint Arch. p. 688 (2013)
48. Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory Symposium– ANTS 2006. pp. 510–524. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
49. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2009)
50. Yang, K., Chen, L., Zhang, Z., Newton, C.J., Yang, B., Xi, L.: Direct anonymous attestation with optimal TPM signing efficiency. IEEE Transactions on Information Forensics and Security **16**, 2260–2275 (2021)
51. Zhang, X., Lin, D.: Analysis of optimum pairing products at high security levels. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. pp. 412–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
52. Zhao, C.A., Xie, D., Zhang, F., Zhang, J., Chen, B.L.: Computing bilinear pairings on elliptic curves with automorphisms. Designs, Codes and Cryptography **58**(1), 35–44 (Jan 2011)